UNIVERSITY OF TRENTO - Italy

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE

ICT International Doctoral School

Learning From Noisy Data Through Robust Feature Selection, Ensembles And Simulation-Based Optimization

Andrea Mariello

Advisor

Prof. Roberto Battiti

University of Trento - Italy

March 2019

Alla mia nonna,
la mia guida invisibile.

To my grandmother,
my invisible guide.

Abstract

The presence of noise and uncertainty in real scenarios makes machine learning a challenging task. Acquisition errors or missing values can lead to models that do not generalize well on new data. Under-fitting and over-fitting can occur because of feature redundancy in high-dimensional problems as well as data scarcity. In these contexts the learning task can show difficulties in extracting relevant and stable information from noisy features or from a limited set of samples with high variance. In some extreme cases, the presence of only aggregated data instead of individual samples prevents the use of instance-based learning. In these contexts, parametric models can be learned through simulations to take into account the inherent stochastic nature of the processes involved. This dissertation includes contributions to different learning problems characterized by noise and uncertainty. In particular, we propose i) a novel approach for robust feature selection based on the neighborhood entropy, ii) an approach based on ensembles for robust salary prediction in the IT job market, and iii) a parametric simulation-based approach for dynamic pricing and what-if analyses in hotel revenue management when only aggregated data are available.

Keywords

Feature selection, Ensembles, Simulation-based optimization, Mutual information, Revenue management

Acknowledgments

I would like to express my sincere gratitude to my advisor Prof. Roberto Battiti, for his support during the PhD. His guidance and experience helped me in improving my research skills and becoming more autonomous.
I would like to thank also the members of the LION lab (past and present), in particular Prof. Mauro Brunato, Dr. Tahir Kalaycı and Manuel Dalcastagnè, for the fruitful discussions throughout my research. I extend my gratitude also to Prof. José Alberto Hernández of the Carlos III University of Madrid for the productive collaboration, to the professors of the courses that I attended during my PhD, and to the administrative staff of the PhD school for clearing doubts on regulations and procedures.
I would like to thank also Prof. Laetitia Jourdan, Prof. Vladimir Grishagin, Prof. Ignacio de Miguel, and Prof. Rodolfo Baggio, for their valuable feedback and time spent on reviewing this dissertation.
My thanks also goes to my friends in S. Bartolameo residence, at the University of Trento and at the IMDEA Networks Institute, for the wonderful time spent together. In particular, I would like to thank Ngô Chấn Nam for the help, support and closeness during my stay in Trento.
A special thanks also to my family, for their continuous and unconditional love and support during my PhD and in my life in general. Last but not least, a heartful thanks to Thủy (and her family). She is the light in the dark that guides me in tough times.

Contents

List of Tables

List of Figures

Publications originated from this work

The research work presented in this PhD dissertation led to the following publications:

- A. Mariello and R. Battiti. Feature Selection Based on the Neighborhood Entropy. *IEEE Transactions on Neural Networks and Learning Systems*, 2018, doi:10.1109/TNNLS.2018.2830700.

  Abstract: In this work we propose a new measure for feature selection that is related to Mutual Information, called Neighborhood Entropy, and a novel filter method based on its minimization in a procedure involving approximated nearest-neighbors and locality-sensitive hashing. Experiments show that the classification accuracy is usually higher than that of other state-of-the-art algorithms, with the best results obtained with problems that are highly unbalanced and non-linearly separable.

  Related thesis chapter: Chapter 2.

- I. Martín, A. Mariello, R. Battiti, and J. A. Hernández. Salary Prediction in the IT Job Market with Few High-Dimensional Samples: A Spanish Case Study. *International Journal of Computational Intelligence Systems*, 11(1):1192–1209, 2018, doi:10.2991/ijcis.11.1.90.

  Abstract: The explosion of the Internet has deeply affected the labour market. Identifying the most rewarded and demanded items in job offers is key for recruiters and candidates. This work analyses 4000 job offers from a Spanish IT recruitment portal. We conclude that experience is

more rewarded than education, we identify five profile clusters based on required skills, and we develop an accurate salary-range classifier by using heterogeneous voting ensembles.

Related thesis chapter: Chapter 3.

- A. Mariello, M. Dalcastagnè, M. Brunato, and R. Battiti. HotelSimu: a Parametric Simulator for Hotel Dynamic Pricing. *Simulation Modelling Practice and Theory* [under review, 2nd round].

  Abstract: In Hotel Revenue Management, an optimal pricing policy is crucial for maximizing the profit. In this work we propose a hotel demand simulator, HotelSimu, which generates events by using Monte Carlo simulations and parametric demand models. These models are completely defined by a limited number of manager-friendly indicators, and they cover a large range of demand curve profiles. Our models can be calibrated by using historical records but at the same time they allow hotel managers to perform fast what-if and risk analyses on future scenarios, even when only aggregated data are available. The applicability of our models within a simulation-based revenue optimization context is evaluated in a case study involving 10 hotels in Trento, Italy.

  Related thesis chapter: Chapter 4.

Other related work during the PhD led also to the following publication:

- R. Battiti, M. Brunato, and A. Mariello. Reactive Search Optimization: Learning While Optimizing. *Handbook of Metaheuristics*. International Series in Operations Research & Management Science, Springer, Cham, 272:479–511, 2019, doi:10.1007/978-3-319-91086-4_15.

Chapter 1

Introduction

In an era characterized by large volumes and variety of data, one possibility to extract useful knowledge is represented by machine learning. The idea that a computer can learn patterns and regularities from raw data without being explicitly programmed has found applicability to almost every aspect of our society. Autonomous systems, speech synthesis, sentiment analysis, predictive maintenance and smart cities are just a few examples of applications of machine learning that are becoming part of our every-day life. However, this ubiquitous presence leads also to serious threats related to the reliability of these approaches in contexts characterized by high uncertainty. In this Chapter, we introduce the problem of learning in the presence of noise and some of the most effective solutions proposed in the literature. We also highlight our contributions related to three scenarios: learning from data with redundant features, learning from a limited number of samples, and learning from aggregated data. We conclude with the outline of this dissertation.

1.1 Learning from Data

At the intersection of Statistics, Computer Science and Operations Research, Machine Learning (ML) addresses the problem of the definition and optimization of a computational system from a set of raw data. ML follows a different

paradigm with respect to other traditional computational approaches. For techniques that do not learn from data, an algorithm is implemented as a sequence of explicitly programmed instructions. ML algorithms instead do not require the expertise of the practitioner to identify patterns and regularities in a dataset. Once these patterns are learned, they can be used to infer knowledge on new instances.

Nowadays, almost all new technologies and companies are data-driven and employ ML approaches. When a social media platform suggests friends appearing in pictures or a smartphone is unblocked with our eyes, we use ML for facial recognition. When a virtual assistant (or chatbot) interacts with us to provide support in choosing the best product or finding directions to reach our destination, we use ML for natural language processing, speech analysis and speech synthesis. When a document scanner transforms images of text into editable words, we use ML for optical character recognition (OCR). When online services suggest the next product to buy or the next movie to watch according to our preferences, we are the target of ML-based recommendation systems. When we communicate through optical networks, ML is employed in the characterization and operation of network components as well as in performance monitoring and estimation of the quality of transmission [96]. Since ML is a continuously developing field, the number of such applications is destined to grow.

Nonetheless, ML algorithms can be classified into two main categories: supervised learning and unsupervised learning. In supervised learning a model is induced from a set of samples expressed as input-output pairs. From these data a mapping is retrieved from the attributes that characterize the instances (the input features) to their corresponding target (the output variable). The learning process relies on the available outputs to reduce the error between the ground truth and the predictions of the model. In particular, when the model learns to predict a category, class or label, we refer to it as a

classification task. When the model is asked to predict a numerical output, we refer to it as a regression task. In unsupervised learning, the samples do not have an associated target. In this case the main goal is to learn groups or clusters of data characterized by common features. Another related class of problems is semi-supervised learning, where the targets of some samples are available (usually for a small fraction of the data), while for most of the samples outputs are not available. In these types of problems, the instances with unknown targets can be used to improve the predictions made by the learners trained through supervised techniques. Another category of ML algorithms is also reinforcement learning, where models (or action policies) are learned, in supervised as well as unsupervised settings, through a dynamic process of trial and error. Most of the times an agent learns the set of the best actions to take in response to specific events in order to maximize a final reward.

The implicit assumption in designing and developing most of the aforementioned ML models is the presence of a noise-free environment. However, in real-world applications ML algorithms should account for different degrees of uncertainty, noise, missing information, errors and imbalance, which can affect the accuracy of the learned models, as reported in the next Section. In this dissertation we analyze the effect of uncertainty in three different application domains, with a main focus on classification, regression and reinforcement learning.

1.2 Learning in the Presence of Noise

Research on ML algorithms has frequently focused on controlled environments and noise-free scenarios [95] because learning accurate models in contexts characterized by high levels of noise and uncertainty can be a challenging task. In real scenarios, the presence of acquisition errors as well as

the absence of some data can significantly affect the generalization performance of the learned models. The training process can also be deceived by redundant or irrelevant features, by a limited number of samples, or by the intrinsic stochastic fluctuations of the analyzed phenomena.
Sometimes the sensing phase, in which the algorithm encodes physical events into machine-interpretable inputs, is not considered explicitly. A frequent hypothesis is the presence of well-defined and already encoded samples that can be used directly for training a model. As an example, in the game of battleship an ML algorithm can be trained to choose the best next move given the state of the board. The procedure would probably use the common encoding of moves as a combination of a letter and a number (e.g. *D-2*). To limit the complexity of the problem, the algorithm would probably concentrate on the reasoning about the game and would avoid considering possible errors in reading or typing the moves. However, to play a real game, the algorithm should account for sensing noise explicitly, and it should distinguish between acceptable and illegal moves. Games are usually characterized by a set of rules to follow but in other application domains a clear definition of what is acceptable is not available. As an example, in the context of anomaly detection, sometimes the definition of anomaly is the result of subjective judgment because a ground truth is missing.
In contexts where there are no prescribed rules, it is also more difficult to clearly define and separate the noise, which should lead to data cleansing or removal, from the concepts of novelty and uncertainty, which are also related to unexpected or unpredictable variation in data but they are not always harmful. The robustness to unexpected modifications in the input is even more important when the ML algorithm is one of the modules of complex data-driven workflows, where cascade effects can compromise the quality of the final products. Whenever noise is expected, attention has to be paid also on the transformations applied to data. As an example, the discretization or

quantization of a numerical attribute with a continuous domain can increase the level of noise. Errors in the original attribute not changing the discrete value or quantized level vanish and do not affect learning. On the other hand, small changes close to the quantization-level boundaries can lead to potentially large qualitative changes in the learned model [108].

1.2.1 Effects of Noise on Machine Learning

Several studies have recently focused on the effects of injected noise on ML algorithms. For example, in [100] the authors show that the Naïve Bayes classifier is systematically more robust than decision trees and support vector machines. Other studies in computer vision show that convolutional neural networks tend to be very sensitive to noisy, distorted and blurred images [77, 46]. Classifiers and in particular deep learning models often suffer from integrity attacks. In [103] a deep neural network is fed with adversarial samples, which are built by adding to legitimate inputs enough perturbations to mislead the model, while a human observer would still perform a correct classification. Experiments show that these adversarial instances lead to a classification error of $\approx 84\%$. Even pre-trained and state-of-the-art models like the Google Cloud Vision API can be seriously affected by noise and adversarial examples [71]. These results raise also security issues for real-world application domains, such as banking, healthcare, and autonomous driving. For example, a face recognition system for online banking can be deceived into granting access to an intruder with traits similar to the owner of the account, or an autonomous vehicle can wrongly identify street signals in rainy weather.

To address these issues, one has to focus on two main approaches:

- *robust learning*, in which traditional ML algorithms are modified to explicitly take into account the presence of noise and gracefully degrade

their performance as the uncertainty grows;

- *noise detection and cleansing*, in which a preprocessing step is executed before the actual learning to identify noisy data and remove or correct them.

Most of the times these approaches need to be combined. As concerns image processing, complete architectures based on simultaneous de-noising, deblurring and classification show promising results [43]. The proposed solutions use filters able to project the input images into the space of natural images. Natural images are in fact characterized by specific features, such as high correlation among adjacent pixels, that usually are not preserved in perturbed inputs. Therefore the effect of noisy and adversarial samples can be effectively reduced by ad-hoc feature transformations and projections.
Other approaches to make these algorithms more robust include also regularization and data augmentation [113, 47]. By adding penalties on the magnitude of the learned parameters or by injecting artificial perturbations in the dataset, the model can be less prone to overfitting and it can exhibit better generalization performance.

In the context of decision tree learning, another strategy to reduce overfitting is pruning leaves and branches that can originate from noisy data. Uncertainty and unexpected perturbations usually lead to deeper trees, and reducing the depth of the model can be an effective technique to avoid memorizing the noise. However, the effort in making an algorithm more robust is not always beneficial, maybe because the chosen algorithm is inappropriate for the specific problem at hand. For instance, classical results [54] show that fully-connected neural networks handle numerical data with noise generally better than the ID3 decision trees [109]. With a large number of samples and an artificial additional noise level of 25%, back-propagation leads to an increase in classification accuracy of $\approx 18\%$ with respect to ID3.

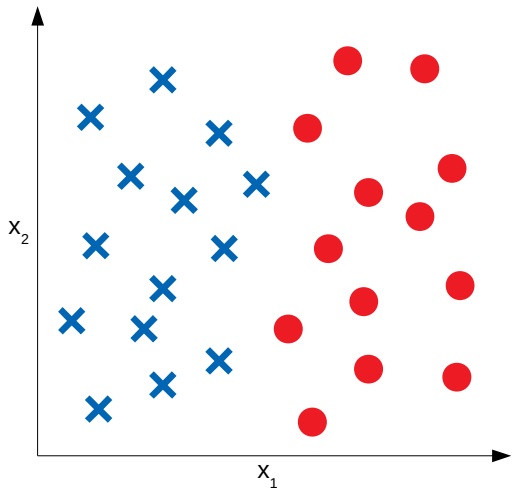


Figure 1.1: Example of a 2D dataset with 2 classes (crosses and circles). The feature x_1 can be used to classify samples without the feature x_2, which seems independent of the class and not relevant for classification.

1.2.2 Noise Reduction by Feature Selection

Despite the fact that noise is usually associated with errors and uncertainty in the samples, the concept can also be extended to the uncertainty in predictions caused by redundant or irrelevant features. A robust algorithm should cope with missing or erroneous information as well as with deceptive information, which can mislead the training of the model and result in noisy predictions and poor generalization performance. A 2D example is reported in Figure 1.1, where one feature is apparently independent from the class and the other feature is more relevant for classification. It is evident that one of the features can be considered as noise with respect to the training process. Several feature selection and feature extraction algorithms have been proposed to reduce the dimension and complexity of the models, and to achieve better prediction accuracy. However, the performance of traditional algorithms can significantly decrease in the presence of noise. As an example, in the context of gene selection, the sensitivity to label noise is relatively high because of the limited number of training samples (usually in the order of tens) [143, 114]. Only a few errors in the output variable lead to a decrease

in the accuracy of feature rankings and of the identification of the most discriminative genes.

One example of a domain-independent approach that implicitly considers noise and stochastic fluctuations is the χ^2 test for stochastic independence [108]. If the hypothesis of independence of a feature from the output variable can be rejected by the test with a high confidence level, the feature is selected for training. The same study also proposes to use the concept of attribute importance, which is defined as the prediction error resulting from the model trained without the considered attribute.

Another well-known relevance criterion is the maximization of the mutual information between the features and the output variable. However, the accuracy of the estimation of mutual information can easily decrease as the level of noise grows. In [88, 55] a more robust estimation is achieved by defining an explicit probabilistic model of the output noise. However, such models are not always available and they do not consider attribute noise explicitly.

1.2.3 Noise Reduction by Ensemble Learning

When the definition of noise models is not an option, an approach to reduce uncertainty and the effects of noisy samples on the final predictions is given by group learning or ensembles. The performance of a group or committee of models is usually better than that of each individual member [44]. The reason behind this is threefold. From a statistical point of view, an ML algorithm searches for the best hypothesis on the data within a hypothesis space. When the search space or dimensionality of the problem are large with respect to the number of samples, it is possible to learn several models with comparable performance but associated with different hypotheses. By aggregating the predictions of a group of models one can reduce the risk of selecting the wrong hypothesis, as depicted in Figure 1.2. From an optimization point of view, training a model can be seen as the optimization of

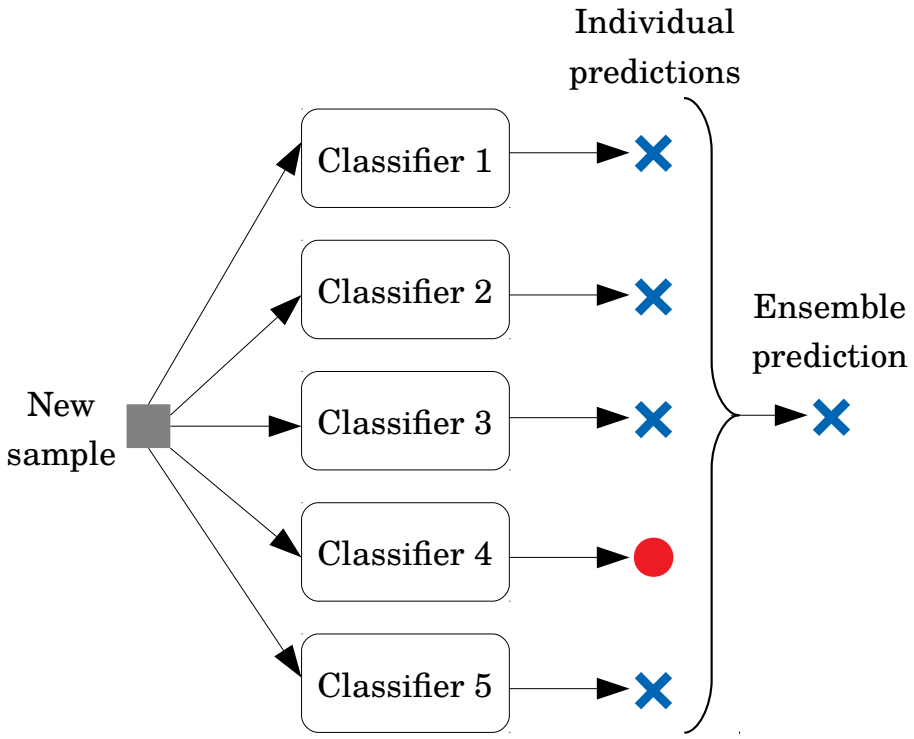


Figure 1.2: Example of an ensemble of 5 classifiers. The risk of an incorrect prediction is reduced by using majority voting among the members of the ensemble.

a cost function. If this function is non-convex, in general a unique global optimum is not available and an optimization algorithm converges to a local minimum. When multiple local minima are present, the risk of selecting a sub-optimal one can be reduced by executing several optimization runs with different starting points. In the context of learning, this is equivalent to training a group of models. Moreover, group learning can be beneficial in terms of the representation capability of the model, since it is more difficult for an individual learner to approximate complex predictor functions like the ones that can be learned by an ensemble.

Several methods have been proposed to build ensembles, including randomization, bagging and boosting [45, 98, 2]. Experiments show that, in situations with low levels of noise, boosting leads to more accurate results than randomization and bagging. When the levels of noise are higher, the variance of the predictions increases. Therefore randomization and bagging, which

are variance-reduction techniques, are usually more robust than boosting. In boosting models, the increase in the weights of instances associated with prediction errors occurs without considering whether the samples have a correct output or not. Putting more emphasis on noise-free samples whose output is difficult to predict can lead to a reduction in the model bias. However, trying to learn the features of noisy inputs and outliers can mislead the learning process.
An alternative approach to bagging and randomization that can lead to robust ensembles is sub-sampling [140]. The main idea of this method is the reduction in the number of instances selected in each bootstrap sample. With lower sampling ratios with respect to standard bagging, the members of the ensemble are trained on more diversified sample sets and can lead to better generalization.
Robust ensembles can also be built as a heterogeneous committee, in which individual members belong to different families of ML models. The final predictions are the result of the aggregation of individual outputs, usually after majority voting or averaging. This approach has the potential to lead to more accurate models because it exploits the distinctive features of each base learner to possibly recognize and handle different types of noise.

Ensembles can also be used as a noise detection procedure, in which a sample is marked as noise when there is disagreement between a specified number of individual members and the ground truth. Similarly to cross-validation, one can split the dataset to be cleaned in different folds and train the ensemble on all the folds but one. The learned ensemble is used to identify noisy instances in the held-out fold, and the same procedure is then repeated on the other folds. In [29], the authors use an ensemble of three classifiers (a decision tree, a linear model and a nearest neighbor classifier), and they partition the dataset into four folds. They also propose two different strategies for identifying noise: majority filtering and consensus filtering. For

the former, they mark an instance as noise when at least two members of the ensemble misclassify it. For the latter, they identify an instance as noise only when all the classifiers disagree with the ground truth, thus leading to a more conservative rule. In [78], a similar approach is used with an ensemble of 25 heterogeneous models. The authors show that the increased diversity among the individual learners effectively reduces the risk of false positives. Other examples of ensembles are presented in [3, 38, 59, 93, 123, 40, 21].

1.2.4 Noise Handling in Imbalanced Data

In addition to ensembles, other interesting approaches for data cleaning include traditional statistical methods for identifying outliers as well as methods based on nearest-neighbors smoothing [14] and disagreement among locally-trained SVMs [111]. However, the previous data cleaning methods are not always sufficient to enhance the quality of the training set. For instance, in classification tasks it is particularly challenging to handle noise in case of imbalanced data, when samples belonging to one class significantly outnumber samples of other classes. This is common in many real scenarios where one is interested in identifying important but infrequent events, such as patients affected by a rare disease.

One way to make the learning process more robust is to retrieve more data. However, this is not always a viable option, and a combination of over-sampling and data cleaning is needed to properly handle imbalanced data with class overlapping [16]. In these contexts, the removal of overlapping samples can lead to better-defined class clusters and to simpler models with better generalization performance. One of the proposed approaches includes a first step of over-sampling based on the Smote technique [34], followed by a data cleaning method based on Tomek links [125]. As depicted in Figure 1.3, new minority-class instances are sampled by using interpolation. Then couples of nearest neighbors belonging to different classes are identified

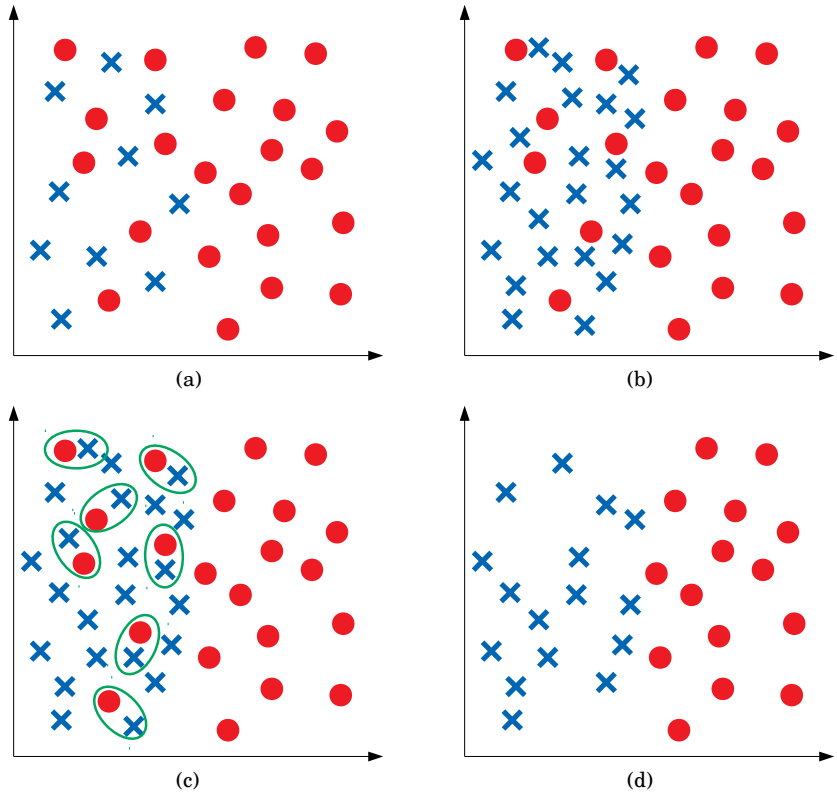


Figure 1.3: Example application of the Smote over-sampling method and of the Tomek links data cleaning method to cure noisy imbalanced datasets. Starting from the original data (a), the minority class is over-sampled (b), and the Tomek links are removed (c). The resulting data (d) are more balanced and with better-defined class clusters.

and removed to obtain better-defined class clusters.

In this dissertation we focus on the effects of noise and uncertainty in the contexts of feature selection, ensembles, and learning through stochastic models, as described in the next Section.

1.3 Proposed Solutions

Among the different application domains in which the effects of noise can be relevant, we address three main scenarios where proper noise handling or modeling can be extremely beneficial for learning. In particular, the main objective of our research is the analysis of uncertainty for problems charac-

terized by different volumes of data.

The first research direction is related to noise handling in the case of feature selection, when the number of samples is sufficient for learning an accurate model but the dimensionality is too high and some features deceive the training process.

The second direction is related to contexts characterized by a limited number of samples that prevents learning accurate models because of poor data quality. In this case, even when the dimensionality is high, feature selection techniques are not always beneficial, and a viable option is given by manual feature engineering and ensemble learning.

The third and last research direction presented in this dissertation is related to the extreme case when individual samples are absent and traditional instance-based ML algorithms cannot be used. Under the hypothesis of having access only to aggregated data, a possibility to extract useful insights is given by the definition of parametric and stochastic models. These models can be used as a baseline or bootstrap sample to infer interesting patterns in real-world phenomena that are difficult to sense and measure.

In more detail, the main contributions of our research include the following. As concerns the feature selection scenario, we propose a novel and robust approach based on the maximization of the mutual information between the features and the output variable. To avoid the errors in evaluating mutual information in noisy and high-dimensional contexts, we define a new measure, which we refer to as neighborhood entropy, that is more robust to noise and class imbalance. Our proposed algorithm selects the features that minimize this measure, by using a greedy procedure based on approximated nearest neighbors and locality-sensitive hashing. Experiments show that the accuracy of models trained with the features selected by our method is usually higher than that caused by other state-of-the-art algorithms, with the best results obtained with problems that are highly unbalanced and noisy.

For scenarios characterized by high dimensionality and a limited number of samples, we propose ensemble learning as a solution to handle uncertainty, increase the robustness of predictions, and improve generalization performance. The effectiveness of our approach is empirically shown in the context of salary prediction in the IT job market. After a rigorous preprocessing step of raw data crawled from the web, we evaluate the benefits of using feature selection, showing that dimensionality reduction does not lead to any advantage with respect to the use of the complete set of features. For this particular application, there is no evidence of redundancy among the features. To increase the prediction accuracy, we formulate the original regression problem as a classification problem for the prediction of a salary range. In addition, we develop and compare several ML models based on linear classifiers, support vector machines, neural networks, nearest-neighbors models and ensembles of decision trees like random forests and boosting machines. We also propose two voting classifiers defined respectively as an ensemble of all the aforementioned models and as an ensemble of the three best performing models. In some cases, this results into an ensemble of ensembles. Experiments show that group learning leads to better accuracy, with the best results achieved by the voting classifiers.

Last but not least, we analyze the problem of learning a model when individual sample instances are not available, and one has access only to aggregated data. There are several real-world scenarios where traditional instance-based ML is not applicable because historical data are not available or are difficult to retrieve. For example, in data storage systems, after some time (years or even hours), the volume of information to be archived is so big that aggregation is used to reduce resource consumption. In this dissertation, we focus on hotel revenue management and in particular on dynamic pricing of hotel reservations. We investigate the case of absence of historical records about reservations and consequently lack of a ground truth for revenues.

Under the hypothesis of known aggregated data like the average number of reservations and cancellations, we propose a set of parametric and stochastic models for the simulation of room demand. We use fixed average prices from 10 hotels in Trento, Italy, as a baseline, and then we optimize our pricing models by using simulation-based black-box optimization. In this case, the learning process is formulated as an optimization procedure, in which the evaluation of the objective function corresponds to a set of Monte Carlo simulations. By following this procedure, one is able to conduct what-if and risk analyses with respect to different scenarios and pricing policies, only with aggregated data. In addition, our methodology explicitly defines the inherent stochasticity of reservation arrivals. Experiments show that the risk of losses is absent for medium-big hotels and limited for small hotels, with a maximum loss probability of ≈ 0.03.

1.4 Structure of the Thesis

The structure of the remainder of this dissertation is as follows.

Chapter 2 presents solutions to the problem of learning from data characterized by redundant or irrelevant information. The main focus is on feature selection for classification, as described in Section 2.1, and in particular on distribution-independent filter methods, which are introduced in Section 2.2. In Section 2.3 we propose a new measure that is related to mutual information, called neighborhood entropy, and a novel filter method based on its minimization. The use of approximated nearest-neighbors and locality-sensitive hashing to speed up the estimation of this measure is described in Section 2.4. As reported in Section 2.5, experiments show that our technique can be employed effectively when one can dedicate more CPU time to achieve better classification accuracy and more robustness to noise and to class imbalance.

Chapter 3 introduces ensemble learning as an effective solution to learn a more reliable model when the number of samples is limited and the problem is characterized by a high dimensionality. In Section 3.1 we describe the application context, which is salary prediction in the IT job market. In Section 3.2 we illustrate our feature engineering strategy to reduce the effect of noise as well as to properly encode and standardize categorical features. In Section 3.3 we define several classification models, including neural networks, random forests, support vector machines, boosting models and ensembles of them. As reported in Section 3.4, experiments show that our dimensionality reduction and data cleansing strategies, coupled with a voting ensemble, lead to an accuracy of $\approx 84\%$.

Chapter 4 tackles the problem of learning a model in the extreme case when individual samples are not available and one has access only to aggregated data. The chosen application domain, which is dynamic pricing in hotel revenue management, is introduced in Section 4.1. An overview of the hotel reservation process and of the proposed simulator is given in Section 4.2. A detailed description of our solution based on parametric models of reservations and cancellations is presented in Section 4.3. Implementation details of the chosen pricing policy and acceptance probability model are then reported in Section 4.4. Experiments on aggregated data from 10 hotels in Trento, Italy, show that the adoption of optimized pricing policies based on our parametric and stochastic models leads to an average revenue increase of $\approx 19\%$ with respect to policies with fixed prices, and to low risk of losses, as reported in Section 4.5.

Chapter 5 concludes the dissertation with some final remarks.

Chapter 2

Robust Feature Selection

When the dimensionality of the problem is high with respect to the number of samples, learning an accurate and robust model is a challenging task. Feature selection is needed to reduce complexity and filter out redundant or irrelevant features that can be considered as noise and therefore deceive the training process. In this Chapter, in the context of classification, we propose a new measure that is related to mutual information, called neighborhood entropy, and a filter method based on its minimization. Our greedy algorithm is also based on approximated nearest-neighbors techniques and locality-sensitive hashing (LSH) to speed up the evaluation of the proposed measure.

Section 2.1 introduces feature selection and the main related algorithms. Section 2.2 focuses on distribution-independent methods and in particular on feature selection based on mutual information. Section 2.3 presents our approach, while Section 2.4 focuses on the use of LSH for finding nearest neighbors. Section 2.5 reports our experimental results. We show that the classification accuracy, for models learned on the features selected by our procedure, is usually higher than that of other state-of-the-art algorithms. At the expense of more computation time, our approach leads to more robustness to noise and class imbalance, and to a better order by which the features are selected, with higher accuracy for fewer features.

2.1 Taxonomy of Feature Selection Methods

When using huge collections of data in classification tasks, one faces the so-called *curse of dimensionality*: the number of dimensions can be too high to get a good classifier in a reasonable time. In this context, dimensionality reduction can lead to:

- better generalization by using a smaller number of free parameters;
- avoidance of the so-called *peaking phenomenon* [80], that is, the situation in which the classification accuracy steadily grows as the number of features grows until a point is reached when adding new dimensions to the system makes the classification accuracy stable or even worse;
- better understanding of the learned models, by making them more interpretable by humans [19].

Dimensionality reduction can be achieved by *extracting* new features (e.g. through projections [129, 130, 27]) or by *selecting* a subset of the original features. In order to select the most relevant features with respect to the output class, several methods have been proposed. By considering the relationship between feature selection and classification algorithms, the various methods can be classified into:

- *wrapper* methods, which use a model and its performance to rank the features [82];
- *embedded* methods, which implicitly select features during the model training [124, 101, 72, 133];
- *filter* methods, which rank and select features by using a proxy measure (*relevance* score) instead of the classifier error rate.

Filter methods are independent from the classifier and are usually faster than wrapper methods, so that they can easily scale to very high-dimensional datasets. The relevance score used by filter methods is computed according to different criteria, including:

- statistical tests like the t-test [74];
- the Fisher linear discriminant [121];
- the Pearson correlation coefficient [63, 136];
- the class of the nearest neighbors [79, 83];
- the mutual information [18, 30, 49, 126, 27].

The methods based on the correlation coefficient capture linear relationships between random variables but they cannot deal with non-linear relationships. The methods based on the t-test and the Fisher linear discriminant require that the features originate from a normal probability distribution. The mutual information (MI) measures arbitrary dependencies between random variables, and it is suitable for assessing the *information content* of features in complex classification tasks. Moreover, the MI measure does not depend on the particular machine learning model or choice of coordinates, as reported in the next Section.

2.2 Distribution-Independent Filter Methods

Before the detailed description of two algorithms that do not make any assumption about the data distribution, we fix the notation and introduce some useful definitions.

N represents the number of samples (points or patterns) in a dataset and D the number of features (or dimensions). The set of samples is represented as a matrix $\mathcal{F}$ of size $N \times D$. A class (category or label) is assigned to

each sample with a value belonging to the set $\{1, 2, \ldots, N_c\}$. All labels are collected in a *class vector* C of size N. The d-th column vector of size N from $\mathcal{F}$, $1 \leq d \leq D$, is the *feature vector* F_d.

We use the simplified notation $\Pr(x)$ to indicate the probability of X being equal to x, that is, $\Pr(X = x)$, and $\mathrm{Fr}(x)$ to indicate the corresponding sample estimator.

2.2.1 RELIEF and RELIEFF

RELIEF [79] is a feature selection algorithm for binary or continuous input data in classification tasks. The only hypothesis made by the algorithm is that similar samples tend to belong to the same class. This assumption underlies most machine learning techniques, for different measures of similarity. RELIEF assigns a weight to each feature and initializes the D-dimensional weight vector W to zero. Then the following procedure is repeated an arbitrary number of times:

1. select at random a sample X from $\mathcal{F}$;
2. find the nearest neighbor that belongs to the same class of X (the *near-hit* of X, $h(X)$);
3. find the nearest neighbor that belongs to the class that is different from the class of X (the *near-miss* of X, $m(X)$);
4. update the d-th weight in W as follows:

$$W_d \leftarrow W_d - (X_d - h_d(X))^2 + (X_d - m_d(X))^2.$$

The weight of a feature increases if the corresponding component of the near-hit is closer than that of the near-miss and decreases otherwise. When the procedure ends, one can rank the features according to the weight vector and select the ones corresponding to the s biggest weights.

An improved version of RELIEF is RELIEFF [83], which is an extension to multi-class problems that updates the weights with the average contribution of k near-hits and k near-misses for each of the classes that are different from the class of the random sample. As concerns the near-misses, it averages the contribution of each class, weighted with the sample estimator of its prior probability.

2.2.2 Feature Selection based on Mutual Information

This Section presents some of the most successful approaches based on the mutual information, which is related to the concepts of entropy and conditional entropy in information theory.

Definition 2.1 (Entropy)**.** Given a classification task, denote by $\Pr(c), c = 1, ...N_c$ the prior probabilities for the different classes. The uncertainty in the classes is measured by the entropy:

$$H(C) = -\sum_{c=1}^{N_c} \Pr(c) \log \Pr(c). \tag{2.1}$$

Definition 2.2 (Conditional Entropy)**.** Let $\mathcal{D}_d$ be the set of all possible values that the d-th feature can assume, $1 \leq d \leq D$, and call $\mathcal{D}_C$ the set of the different classes. The uncertainty in the classes after knowing the d-th feature is the conditional entropy, which is defined as follows:

$$H(C|F_d) = -\sum_{f \in \mathcal{D}_d} \Pr(f) \sum_{c=1}^{N_c} \Pr(c|f) \log \Pr(c|f). \tag{2.2}$$

The conditional entropy is less or equal to the entropy. It is equal if and only if the feature and the output class are independent random variables.

Definition 2.3 (Mutual Information)**.** The mutual information (MI) between the class C and a feature F_d is defined as follows:

$$\mathrm{I}(C; F_d) = \sum_{c \in \mathcal{D}_C, f \in \mathcal{D}_d} \Pr(c, f) \log \frac{\Pr(c, f)}{\Pr(c) \Pr(f)} \,. \tag{2.3}$$

The mutual information is symmetric in C and F_d, and it can also be seen as the amount by which the entropy decreases when the feature F_d is known:

$$\mathrm{I}(C; F_d) = H(C) - H(C|F_d). \tag{2.4}$$

The MI is related to the accuracy of a learning system because of Fano's inequality [52].

Theorem 2.1 (Fano's Inequality)**.** Let the random variables X and Y represent the input and the output messages of a noisy channel. A receiver operates a function $f(y)$ which, given the received message y, tries to reconstruct the original input x with $\hat{x} = f(y)$. Let E be the binary random variable associated with the event of an error, that is, $E = 1$ when, given the channel's output y, the result of the decoder $\hat{x}$ is different from the original input x. Then the following inequality holds:

$$H(X|Y) \leq H(E) + \Pr(E = 1) \log(N - 1), \tag{2.5}$$

where N is the number of different channel inputs.

According to the so-called *Infomax Principle* [91], by equating Y to a feature F_d, X to the class C, and the function operated by the receiver $f(\cdot)$ to a classifier, then $\Pr(E = 1)$ can be seen as the classification error rate. By (2.5), this probability has a lower bound that is proportional to the conditional entropy $H(C|F_d)$. Therefore, reducing this quantity, which corresponds to increasing the MI, is a necessary condition to lowering the error rate and building an effective classifier. However, this condition is not

sufficient, because one needs a machine learning model capable of *extracting* the available information content.

Several strategies have been proposed for selecting features on the basis of the MI between them and the output variable. The method called Feature Selection based on the MI and the criterion of *Maximum Relevance* (MR-MIFS) [105] computes the value of the MI between the class variable and each individual feature. Then it selects the features with the highest s values. The estimation of the probabilities involved in the computation of the pairwise MI can be done by using histograms as in [18, 30] or Kernel Density Estimators (KDE) as in [86].

Another procedure for feature selection uses the criterion of *minimum redundancy* and *Maximum Relevance*, presented for the first time in the seminal paper [18]. The algorithm starts by selecting the most relevant feature, according to the MI with the class, and then it keeps adding features that are relevant but not redundant, that is, having a high value of MI with respect to the class but a low value of MI with respect to each individual feature that has already been selected. This twofold objective is achieved by maximizing the following quantity:

$$\mathrm{I}(C; F_d) - \beta \sum_{F' \in \mathcal{S}} \mathrm{I}(F_d, F'), \tag{2.6}$$

where β is a weight that regulates the trade-off between relevance and redundancy, and $\mathcal{S}$ is the set of already-selected features. For $\beta = 0$ the algorithm is equal to MR-MIFS.

A drawback of the previous procedure, which we refer to as mRMR-MIFS, is that the second term in (2.6) can increase in magnitude with respect to the first term when the cardinality of $\mathcal{S}$ grows [30]. Therefore, it can become predominant in the choice of the next feature to select. As a consequence, the algorithm also needs a proper value of the β parameter. In [49] the weight of

redundancy is defined as an adaptive parameter that takes into account the cardinality of the set $\mathcal{S}$.
Other methods include approaches based on quadratic divergence [126] and fixed approximated density models [89]. Models derived from Gaussian distributions are used to maximize upper bounds on the MI with the class, without the evaluation of the joint behavior of the selected features.
The idea of maximizing relevance and minimizing redundancy has also been used in methods not based on the MI. A semi-supervised method based on correlation, called RRPC, has been recently proposed in [136]. The main difference from mRMR-MIFS is the maximization of the following quantity:

$$\mathrm{P}(C_L; F_{Ld}) - \frac{1}{|\mathcal{S}|} \sum_{F' \in \mathcal{S}} \mathrm{P}(F_d, F'), \tag{2.7}$$

where $\mathrm{P}(C_L; F_{Ld})$ is the Pearson correlation coefficient between the class and a feature, which is computed only on the labeled data, while $\mathrm{P}(F_d, F')$ includes also the unlabeled data.

One common trait of the previous methods is the pairwise computation of the chosen score, either between a feature and the class or between two features. They account for relevance and redundancy separately. The technique proposed in [30] is instead based on a generalization of the MI to multiple features. The X-MIFS algorithm evaluates the MI between the whole set of selected features and the class, so as to add only those features that are relevant when considered together. The main difference of this approach is that it does not consider two one-dimensional random variables as before (the class variable C and one feature F_d). It computes the MI between a one-dimensional random variable (C) and a multidimensional random variable $\mathcal{S} = (F_{i_1}, F_{i_2}, \ldots, F_{i_m})$, where $0 < m \leq D$, consisting of m distinct features from $\{F_1, F_2, \ldots, F_D\}$. $\mathcal{S}$ assumes values in $\mathcal{D}_\mathcal{S} = \mathcal{D}_{i_1} \times \mathcal{D}_{i_2} \times \cdots \times \mathcal{D}_{i_m}$. To compute the MI according to (2.3), the sum has to be taken over all possible

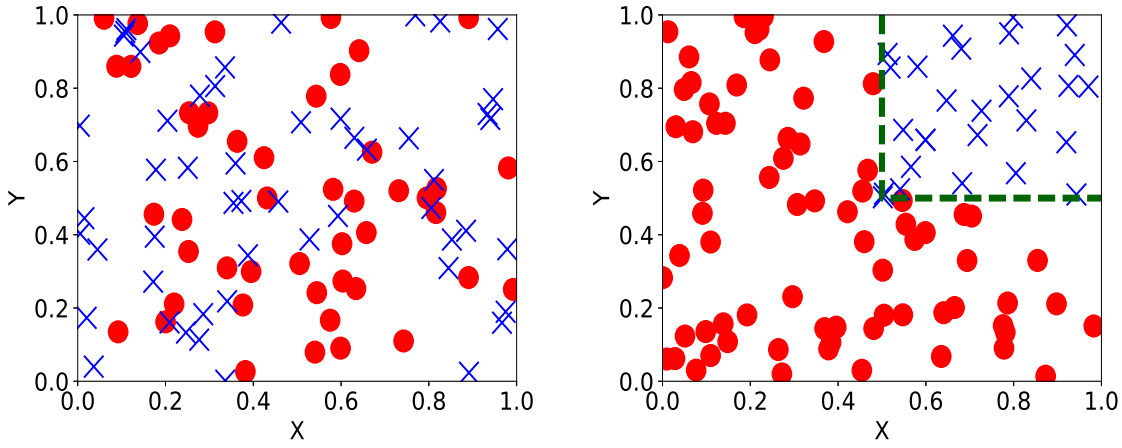
values in $\mathcal{D}_{\mathcal{S}}$. The probability distributions for the computation of the MI as in (2.3) are evaluated by using histograms and the relative frequencies of each single value in $\mathcal{D}_{\mathcal{S}}$. However, X-MIFS requires a large amount of memory to store the frequencies for all the values in $\mathcal{D}_{\mathcal{S}}$. As an example, with binary features the number of these values is equal to $2^{|\mathcal{S}|}$.

Another possibility for a fast evaluation of the MI by using a certain level of approximation and an index with a smaller memory footprint is given in [84]. The MI between two continuous variables is estimated after making the hypothesis that, for a given point, the distribution in its neighborhood (within a sufficiently small radius) remains constant. The authors propose two estimators based on the approximated k-nearest neighbors retrieved from the space identified by the two variables. These estimators can be also used to compute the MI between multiple variables, but they cannot be used for classification, when one needs the MI between a feature (numerical or not) and a categorical class, since there is no concept of distance among categories. As described in the next Section, under the hypothesis of numerical features, we propose a technique for evaluating an MI-related measure with the same idea of approximated neighborhoods. However, the search for the nearest neighbors is performed only on the features, thus making our method suitable for classification.

2.3 Feature Selection with the Neighborhood Entropy

Our proposed algorithm starts by selecting features through a greedy procedure that maximizes the MI, as in X-MIFS. To maximize (2.4), which is equivalent to (2.3), the first term in the equation (the class entropy) can be omitted because it is constant. One is left with the minimization of the conditional entropy $H(C|F_d), 1 \leq d \leq D$. A reasonable assumption for a successful classifier is that similar points should belong to the same class, with

the exception of those points that are on the class boundary. If this assumption holds true for a specific feature, then the uncertainty in determining the class (the conditional entropy) should be very small and the learning process can benefit from considering that feature. On the contrary, when for a specific feature the class entropy is very high, the classification accuracy cannot improve by considering that feature (sometimes it can even get worse).
A toy example is illustrated in Figure 2.1, with 100 points drawn from a uniform distribution defined on the unit square $[0, 1] \times [0, 1]$ and with only 2 classes. In Figure 2.1a each class is randomly assigned to one half of the points. In Figure 2.1b all the points with both of the coordinates that are greater than 0.5 belong to one class, and all the remaining points belong to the other class. In the first case the conditional class entropy is maximum and equal to the class entropy ($\log 2$), and the features (X and Y) cannot be used effectively for the classification. In the second case, the separation surface is completely defined by the two features, which are then relevant for the classification since their conditional entropy is minimum.
A possible strategy for evaluating the class entropy for a specific feature without estimating probability distributions is the following: one considers the class values of the samples in the neighborhood of a point, computes the class entropy only for those points and then takes the average over all the points in the dataset. In the case of very high entropy (as in Figure 2.1a), the average uncertainty computed in this way remains very close to the theoretical value of $\log 2$. In the second case (Figure 2.1b), for most of the points there is no uncertainty in their neighborhood and the average conditional class entropy is very close to 0. Therefore, one can use the conditional class entropy evaluated on the neighborhoods of the points as a relevance score for selecting the most informative features. Even in the situation of non-uniform distributions, the previous observations remain valid if one considers neighborhoods with a fixed number of neighbors instead of a fixed radius.



(a) (b)

Figure 2.1: The Neighborhood Entropy evaluated for the case of high uncertainty (a) is higher than the value retrieved for the case of low uncertainty (b).

After the previous motivations, we define the concept of Neighborhood Entropy as follows.

Definition 2.4 (Neighborhood Entropy)**.** Given a classification task as defined in Section 2.2 and a multidimensional random variable $\mathcal{S}$, $\mathcal{N}(X;k)$ is the set of a point X, which is an instance of $\mathcal{S}$, and its k-nearest neighbors. Then the relative frequency of a class $c \in \mathcal{D}_C$ within $\mathcal{N}(X;k)$ is:

$$\mathrm{f}_k(c, X) = \mathrm{Fr}(c|\mathcal{N}(X;k)) = \sum_{Y \in \mathcal{N}(X;k)} \frac{n(C = c, \mathcal{S} = Y)}{|\mathcal{N}(X;k)|},$$

with $n(\cdot)$ counting the occurrences of an event.
The k-Neighborhood Entropy (NE) of the class variable C with respect to $\mathcal{S}$ is then defined as follows:

$$\mathrm{NE}_k(C;\mathcal{S}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c \in \mathcal{D}_C} \mathrm{f}_k(c, X^i) \cdot \log(\mathrm{f}_k(c, X^i)), \tag{2.8}$$

where X^i represents the i-th instance of $\mathcal{S}$.

Algorithm 1 NEFS: feature selection based on the Neighborhood Entropy (2.8).

function NEFS($\mathcal{F}$,C,k,s)
  $\mathcal{S} \leftarrow \emptyset$
  $\mathcal{R} \leftarrow \{F_1, F_2, \ldots, F_D\}$
  repeat
    find F^* that minimizes $\mathrm{NE}_k(C; \mathcal{S} \cup \{F\}), \forall F \in \mathcal{R}$
    $\mathcal{R} \leftarrow \mathcal{R} \setminus \{F^*\}$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{F^*\}$
  until $|\mathcal{S}| = s$
  return $\mathcal{S}$

Our k has an effect similar to the bandwidth of a kernel density estimator. Choosing a large k corresponds to using wider kernels for approximating the distribution at a point.

In addition, the minimization of the NE is equivalent to the maximization of the MI as in (2.3) or (2.4), since one can approximate the conditional entropy in (2.2) with the following:

$$\mathbb{E}_{\mathcal{N}(\mathcal{S};k)}[H(C|\mathcal{S})] \approx \frac{1}{N} \sum_{i=1}^{N} H(C|\mathcal{N}(X^i; k)) = \mathrm{NE}_k(C; \mathcal{S}).$$

The distinctive feature of the NE with respect to the MI is that the former allows selecting features by maximizing the MI with no need to estimate the feature probability distributions directly. Instead it employs an adaptive estimator of the class uncertainty by focusing only on the class values in a neighborhood of a point.

Our proposed strategy, called *NEFS*, is reported in Algorithm 1. It is based on a greedy procedure similar to the one used by X-MIFS: at each iteration the algorithm selects the feature that minimizes the NE of the output variable with respect to the union of that feature with the already-selected subset. The main difference is in the cost of evaluating the NE ($\mathcal{C}_{\mathrm{NE}}$), which is related to the specific implementation of the nearest neighbor algorithm. If i is the

number of already-selected features at the iteration $i+1$, the CPU time complexity of NEFS for selecting s features can be expressed as follows:

$$\mathcal{C}_{\text{NEFS}} \;=\; \mathcal{C}_{\text{NE}} \sum_{i=0}^{s-1} D - i = O(\mathcal{C}_{\text{NE}}\; Ds). \tag{2.9}$$

The next Section describes some of the most efficient techniques for nearest neighbors search, with particular focus on *Locality-Sensitive Hashing* (LSH). In our implementation, we use LSH, since it is the fastest when one can tolerate a certain level of approximation.

2.4 NEFS Implementation Details

The evaluation of the NE as in (2.8) requires searching for the k-nearest neighbors for each point in the dataset, by using distances computed on spaces of varying dimensions. Before delving into some of the solutions to speed up search queries, let us define the different problems related to searching for nearest neighbors.
Let $\mathcal{F}$ be a set of N points defined on a space $\mathcal{M}$ of D dimensions, with a distance function $dist : \mathcal{M} \times \mathcal{M} \mapsto \mathbb{R}_0^+$.

Definition 2.5 (k-Nearest Neighbors problem (k-NN))**.** Given any point $Q \in \mathcal{M}$, find the points $P_1, P_2, \ldots, P_k \in \mathcal{F}$ that have the smallest, the second-smallest, ..., the k-th smallest distances to Q, respectively.

Definition 2.6 (ϵ-approximated k-Nearest Neighbors problem (ϵk-NN))**.** Given any point $Q \in \mathcal{M}$ and an approximation factor $\epsilon > 0$, find a set of points $P_1, P_2, \ldots, P_k \in \mathcal{F}$ such that $dist(P_i, Q) \leq (1+\epsilon) dist(P_i^*, Q), i = 1, 2, \ldots, k$, with P_i^* being the i-th true nearest neighbor of Q.

Since we are interested in contexts characterized by high dimensionality, and at the same time by enough samples to learn a robust classifier, we

assume to have a dataset with $D < N < e^D$. This corresponds to having more samples than features, with a very loose upper bound that can hardly be reached in real non-trivial feature selection problems (all the datasets used for the experiments except one fall into this category). Under this hypothesis, computing distances for each point is $O(DN)$. If a *min-heap* is used for sorting, building the heap with the N distances and then retrieving the k smallest is $O(N + k \log N)$. It is also required that $k < N$ (usually $k \ll N$). Since $D > \log N$, the asymptotic complexity of the algorithm that solves the k-NN problem is dominated by the term relative to the computation of distances $O(DN)$. However, NEFS needs to find the nearest neighbors of each individual point in the dataset to estimate the NE. Consequently, another kind of problems needs to be solved.

Definition 2.7 (All k-Nearest Neighbors problem (A-k-NN))**.** For each point $Q \in \mathcal{F}$, find its k-nearest neighbors $P_1, P_2, \ldots, P_k \in \mathcal{F} \setminus \{Q\}$.

Definition 2.8 (All ϵ-approximated k-Nearest Neighbors problem (A-ϵk-NN))**.** Given an approximation factor $\epsilon > 0$, for each point $Q \in \mathcal{F}$, find its ϵ-approximated k-nearest neighbors $P_1, P_2, \ldots, P_k \in \mathcal{F} \setminus \{Q\}$.

If one solves an instance of the k-NN problem for each point in the dataset, the A-k-NN problem can be solved in $O(DN^2)$ time. Other algorithms solve the same problem, with a fixed D, in $O(N \log N)$ time by recursively building a tree of neighboring boxes [39, 127]. However, when the number of dimensions is not fixed, these techniques have a worst-case running time of $O((cD)^D N \log N)$, for a dimension-independent constant c.

In our context, which is characterized by noise and uncertainty, one aims at solving the A-ϵk-NN problem, so as to reduce the query time by tolerating a certain level of approximation. Intuitively, retrieving approximated neighbors corresponds to possibly using a larger radius for the neighborhood of a point. Depending on the level of noise, if k is sufficiently small, the retrieved

points can still be good indicators of the class entropy near the query point. In some cases, it would be convenient to revert to some approximated techniques that solve the A-ϵk-NN problem in $O(DN^{\alpha})$ time, with $\alpha \approx 1$.
This is achieved by building an index that groups samples in their approximated neighborhoods, before the actual evaluation of distances. When searching for neighbors, the index should help in discarding most of the samples in approximately constant time. Then distances are computed on the remaining candidate samples. Several indexing methods have been proposed for the NN search, including *Locality-Sensitive Hashing (LSH)* [58]. A hash table is built with the samples in a dataset by using hash functions that ensure, with some probability, that points that are closer to each other collide into the same bucket, while distant points fall into different buckets. Candidate neighbors correspond to the points of the bucket associated with the query point, and the nearest neighbors are searched among them.
Another possibility is to partition the space into regular grids, which work better with uniformly distributed data. Other methods are instead based on clustering to dynamically adapt to the dataset distribution [99]. However, the building time of those indexes grows significantly compared to LSH (described in the next Section), because a clustering algorithm has to be run and convergence can be hard to reach. Other space-partitioning schemes are based on k-dimensional (or kd) trees [115, 10]. kd trees have been shown to speed up the exact NN search in low-dimensional spaces but they can be as inefficient as an exhaustive search in case of high-dimensional spaces [99].
Our algorithm is implemented together with the LSH method, since it is the fastest and provides guarantees on the accuracy of the results, as reported in the next Section.

2.4.1 Locality-Sensitive Hashing

Let us start by fixing the notation and by introducing the main concepts about LSH, as described in the original paper [58].
Let *dist* be a distance function between elements of a set $\mathcal{F}$ of N points defined in a metric space $\mathcal{M}$ of D dimensions, and for any point $P \in \mathcal{F}$ let $\mathcal{B}(P; r)$ denote the set of points that are within a distance r from P, according to *dist*.

Definition 2.9 (Locality-Sensitive Hash functions)**.** A family $\mathcal{H}$ of hash functions $h(\cdot)$ is called (r_1, r_2, p_1, p_2)-sensitive for *dist*, with $r_1 < r_2$ and $p_1 > p_2$, if for any couple of points $P, Q \in \mathcal{F}$:

- if $P \in \mathcal{B}(Q; r_1)$ then $\Pr_{\mathcal{H}}(h(Q) = h(P)) \geq p_1$;
- if $P \notin \mathcal{B}(Q; r_2)$ then $\Pr_{\mathcal{H}}(h(Q) = h(P)) \leq p_2$.

In NEFS, we implement LSH by using a family of data-independent locality-sensitive hash functions that are suitable for Euclidean spaces and are proposed in [41].
These hash functions are defined as follows:

$$h(v) = \left\lfloor \frac{a \cdot v + b}{w} \right\rfloor, \tag{2.10}$$

where w is a positive integer, b is randomly drawn from $[0, w]$ and $a \in \mathbb{R}^D$ is drawn from a Cauchy distribution, in case of a procedure using l_1-norm distances, or from a Gaussian distribution, in case of l_2-norm distances. Since the guarantees provided by LSH hash functions do not change when one uses the l_1-norm instead of the l_2-norm, and one is interested only in an approximated neighborhood, we opted for the l_1-norm distance, which is slightly faster to compute. This can reduce the running time significantly, since one of the most frequent operations of the procedure is the evaluation of a distance.

In the standard implementation of our technique, a new LSH index is built each time the NE between a set of features and the class is evaluated. This is done $O(Ds)$ times as reported in (2.9). However, L new hash functions are generated only once for each iteration of the loop in Algorithm 1, that is, when the cardinality of the set of selected features changes. For each iteration $i = 1, 2, \ldots, s$ the procedure generates L new hash functions. For each function, it generates the scalar b and the vector $a \in \mathbb{R}^i$. This translates into the generation of a number of coefficients equal to $\sum_{i=1}^{s}(i+1)L = O(Ls^2)$. The actual index construction is done every time a new feature is considered. In this case, each iteration requires hashing each point in the dataset L times by using (2.10). The total number of iterations for building the indexes is $\sum_{i=0}^{s-1} D - i$ as in (2.9). Since, for $i = 0, 1, \ldots, s-1$, applying (2.10) is $O(2i+3)$ and the number of hashed values is NL, the total cost of the building step is $O\left(NL\sum_{i=0}^{s-1}(D-i)(2i+3)\right) = O(NLDs^2)$. Once the index has been built, the nearest neighbors for each point in the dataset are retrieved by using the hash values of the points to find the promising candidates in each of the L hash tables. Then the standard nearest neighbor search is performed on the list of candidate points to find the k-nearest neighbors among them. As reported in [58], the time for one query on D-dimensional points is $O(DN^{1/(1+\epsilon)})$.

If $i+1$ represents the number of features on which each index is built for $i = 0, 1, \ldots, s-1$, retrieving the neighbors for one point is $O((i+1)N^{1/(1+\epsilon)})$. Therefore the total cost for solving the A-ϵk-NN problem for estimating the NE for all the subsets of features is

$$O\left(\sum_{i=0}^{s-1}(D-i)N(i+1)N^{1/(1+\epsilon)}\right) = O(N^{(2+\epsilon)/(1+\epsilon)}Ds^2).$$

Consequently, if $\alpha = (2+\epsilon)/(1+\epsilon)$, the computational complexity of NEFS

can be expressed as follows:

$$\mathcal{C}_{\text{NEFS}} = O(Ls^2 + NLDs^2 + N^{\alpha}Ds^2) = O(Ds^2N^{\alpha}). \tag{2.11}$$

The last equality holds because, according to the equations in [58], $L \approx N^{1/(1+\epsilon)}$, given a high probability of collision for neighbors (e.g. 0.9) and a low probability of collision for distant points.

To reduce the time for the NN queries, our procedure evaluates the NE on the neighborhoods of samples not already visited within the neighborhood of another sample, as illustrated in Figure 2.2. The average is then taken only on the visited points. This corresponds to a reduction in the number of estimators from N to roughly N/k. Consequently, the running time of NEFS decreases proportionally to k. The decrease is not equal to k times because the index is still built with all the points in the dataset to exploit as much information content as possible. It is interesting noting that this strategy is similar to the random selection of points in RELIEFF, with the main difference being that the choice of points is not random but guided by the points themselves.

We developed two implementations of NEFS. The first version builds an LSH index each time the NE is evaluated, by using only the current subset of features involved in the NE computation. This strategy has the potential of being very robust to noise because the different indexes can dynamically adapt to the different sub-spaces involved in the NE computation. However, building several indexes can result in more computation time. The second version, which we refer to as *Single-indexed* NEFS (SNEFS), builds the LSH index only once on the D-dimensional space identified by all the features. When the NE on a subset of features is evaluated, at first the procedure retrieves the neighbors of a point in the full-dimensional space. Then a search is done for the nearest neighbors among the candidates, after they have been projected onto the reduced space. This idea originates from the fact that

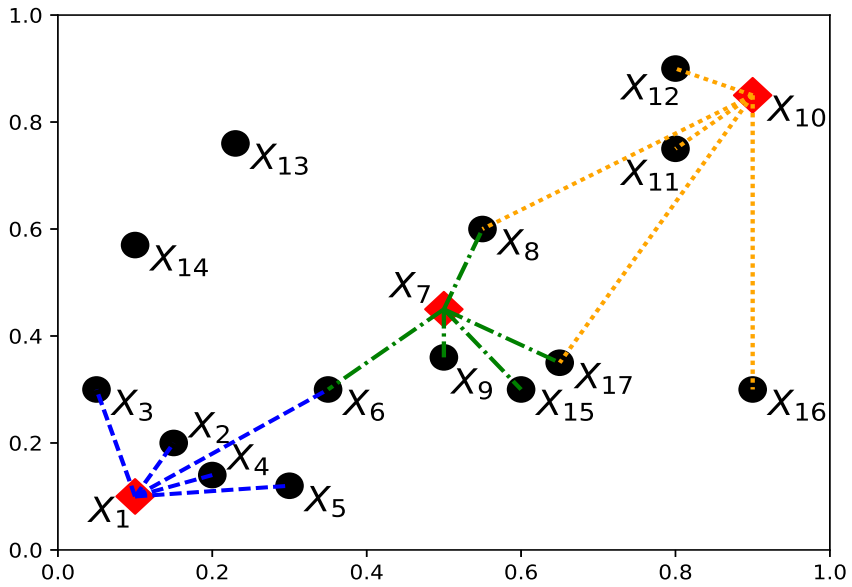


Figure 2.2: First 3 iterations of NEFS on 17 two-dimensional points and $k = 5$. The first iteration selects the nearest neighbors of X_1. Since $X_i, i = 2, \ldots, 6$ have already been considered, the second iteration selects the neighbors of X_7. In this case X_6 is selected again. The third neighborhood is that of X_{10}.

neighbors in D dimensions are also neighbors in s dimensions, with $s < D$, and one does not need the exact neighbors as well as *all* the neighbors for estimating the class entropy near each point. A further code optimization reduces also the time spent for querying the index. While the NN search on the projected candidates has to be done whenever the subset of features changes, the (full-dimensional) candidate neighbors for each point in the dataset do not change and can be retrieved only once. If each candidate point is projected onto $i+1$ features for $i = 0, 1, \ldots, s-1$, the NN search for one point is still $O((i+1)N^{1/(1+\epsilon)})$, with a total cost for solving the A-ϵk-NN problems equal to $O(N^{\alpha} D s^2)$. Accordingly, the complexity of the modified procedure is:

$$\mathcal{C}_{\text{SNEFS}} = O(LD + NLD + N^{\alpha} D s^2) = O(D s^2 N^{\alpha}). \tag{2.12}$$

This modification usually gives better results in terms of classification accu-

racy and, even though the asymptotic complexity remains equal to that of NEFS, it makes the actual running time reduce significantly, as shown in the next Section.

2.5 Experiments and Discussion

In this Section, we compare our proposed approach, with the two different implementations NEFS and SNEFS, to other state-of-the-art filter methods. In particular, we analyze RELIEFF, X-MIFS, RRPC and mRMR-MIFS. The results are related to the classification accuracy of models learned by using the features selected by the different algorithms. For a fair comparison, we use the RRPC method in its original semi-supervised version (RRPC_SS), with 80% of (artificially) unlabeled data, as well as in a completely supervised version (RRPC_S). As concerns mRMR-MIFS, we fix $\beta = \frac{1}{|\mathcal{S}|}$ to be consistent with the choice made by RRPC in (2.7). For each algorithm and dataset, we train and evaluate:

- a Random Forest (RF) of 20 trees, with *bootstrap* and the *Information Gain* split criterion;
- a Support Vector Machine (SVM) with a Radial Basis Function (RBF) kernel;
- a k-Nearest Neighbors classifier (kNN) with $k = 4$;
- a Multi-Layer Perceptron (MLP) with one hidden layer of 20 neurons, the tanh activation function and *early stopping*.

For training and validating the previous models, we used the implementation provided by the Python library *scikit-learn*[104]. The method used for the optimization of MLP weights is *stochastic gradient descent* (SGD), with an adaptive learning rate to reduce convergence time. The technique provided

by the library is similar to the *bold driver* method [17], with the addition of early stopping (ES) to avoid overfitting. At each iteration, 20% of the training set associated with one fold of cross validation is kept apart as an ES-validation set. Starting from an initial value of 1, the learning rate is kept constant as long as the training loss keeps decreasing. Each time two consecutive epochs fail to decrease the training loss or to increase the classification accuracy on the ES-validation set by a tolerance value, the current learning rate is divided by 5. We fix the number of iterations for early stopping to 100 and the tolerance value for improvement in the validation score to 10^{-8}. The tests were conducted on a 64-bit 8-core machine with a CPU frequency of 2.3 GHz and a total of 4 GB of RAM. Only a sequential implementation of all the algorithms was tested. In particular, we developed our own code for NEFS, SNEFS, RELIEFF, RRPC and mRMR-MIFS, while we used the original implementation of X-MIFS developed by the authors in [30].
The GAS benchmark is taken directly from the UCI Machine Learning Repository [90]. Two other datasets (SPAMBASE_N and SEGMENT_N) are modified versions of the same datasets in [90], taken from the KEEL dataset repository [6]. The modified versions add a 20% noise level to the features according to the schema proposed in [146].
We also use the CORRAL synthetic dataset [76], which includes 6 features (A0, A1, B0, B1, I and C). The class is defined by $(A0 \wedge A1) \vee (B0 \wedge B1)$. I is randomly generated, and C is highly correlated with the class, with 25% of error rate. Moreover, another dataset (HYPERSPHERES) was generated to test the algorithms against highly unbalanced binary classes, defined by non-linear relationships with a small subset of features (7 among 100). We generated 100-dimensional samples normally distributed (with zero mean and unit variance) in the space $[-10, 10]^{100}$. A class 1 was then assigned to

Table 2.1: Datasets used for the tests

Name	N	D	N_c
CORRAL	128	6	2
GAS	13910	128	6
HYPERSPHERES	5000	100	2
SEGMENT_N	2310	19	7
SPAMBASE_N	4597	57	2

samples satisfying at least one of the following:

$$((X_6^2 + X_{20}^2 + X_{53}^2 + X_{22}^2 + X_{87}^2) \leq 100) \quad \vee$$
$$(((X_{10} - 8)^2 + (X_{44} + 3)^2 + (X_{53} - 5)^2) \leq 25),$$

where X_i is the value of the i-th feature of the sample X. With this choice of parameters and features, among 5000 generated samples we obtained nearly 1000 belonging to class 1, which is represented by the union of points in two hyperspheres (5D and 3D) with non-empty intersection. In this way we are able to assess the behavior of the feature selection techniques in the case of (known) complex dependencies between the features and the class, within a very noisy context. For each dataset, we removed the samples with missing values and we transformed categorical variables into one-hot encoded variables. The final number of samples, features and classes for each dataset are reported in Table 2.1.

For the tests on the HYPERSPHERES dataset, the classifiers are trained on 7 features, 4 for CORRAL, while for the other datasets 10 features are used. In this way one can see the ability of the algorithms in selecting the best features as soon as possible, independently from the highest level of accuracy that they can reach with a customized threshold. In each case all the classifiers are validated by using a 10-fold cross validation. The number

of neighbors k was fixed to 4 to obtain a more precise approximation of the probability distributions, and the number of iterations of RELIEFF was set to N/k to make it comparable to our technique, after the description in Section 2.4.1. For each dataset and feature subset we assess the statistical significance of the results by using the Friedman test [56]. In the following, our findings are expressed in terms of averages and standard errors and, unless otherwise noted, they are statistically significant with a significance level $\alpha = 0.01$.

In the comparison with CORRAL, the algorithms do not present statistically significant differences in the average accuracy because of the very limited set of samples. However, the techniques based on the nearest neighbors (NEFS, SNEFS and RELIEFF) are able to select the 4 most relevant features most of the times, while the others tend to select the less relevant feature C as the first one, as reported in Table 2.2.

In the case of GAS, the convergence rate is usually higher for SNEFS. As an example, for the MLP there is an increase in accuracy of 5.4% with respect to X-MIFS with 4 features, as illustrated in Figure 2.3a.

In the case of spam detection, the best classification accuracy ($\approx$0.85) is reached by the RF trained on the features selected by NEFS and SNEFS, as reported in Figure 2.3b. This accounts for approximately a 3.7% increase with respect to RELIEFF, 14.9% compared to X-MIFS, and 25% for RRPC and mRMR-MIFS. Our techniques have better results even with one or two features. We also compare the algorithms on random subsets of SPAMBASE_N, to show the robustness to data scarcity of our NN-based estimators. As expected, the estimation error is higher for all the compared algorithms and the classification accuracy is slightly lower. This is due to the reduced amount of samples, which leads to higher variance and generalization error. However, our techniques still lead to generally better results, as reported in Figure 2.3c for $N = D$.

Table 2.2: Most frequently selected features for the CORRAL dataset.

Method	Rank			
	1st	2nd	3rd	4th
X-MIFS	C	A1	B1	B0
RELIEFF	B0	B1	A1	A0
SNEFS	B0	B1	A1	A0
NEFS	B0	A1	B1	A0
RRPC_S	C	B0	B1	A1
RRPC_SS	C	B1	A1	A0
mRMR	C	B0	B1	A1

For SEGMENT_N, in Figure 2.4a it is evident that NEFS and RELIEFF are not as robust as with spam detection, especially with fewer features. SNEFS instead leads to better generalization and less overfitting, because it limits the construction of the index to the set of all the features.

In the case of HYPERSPHERES, none of the algorithms is able to select all the 7 features used to generate the class, as shown in Figure 2.4b. However, SNEFS and NEFS reach significantly higher values of accuracy, since they are able to select 5 or 6 of the features needed in most cases. RELIEFF and RRPC are clearly the worst, since they are not able to go further than 1 or 2 *relevant* features. It is also worth noting that the third feature selected by X-MIFS and mRMR-MIFS corresponds most of the times to the 53rd feature in the dataset, which is at the intersection between the two hyperspheres. This explains why X-MIFS and mRMR-MIFS are slightly better than the others with the first 3 features. However, the effect of the 93 noisy features becomes evident with more than 3 selected features and those methods lead to a classification accuracy up to 12.3% less than NEFS.

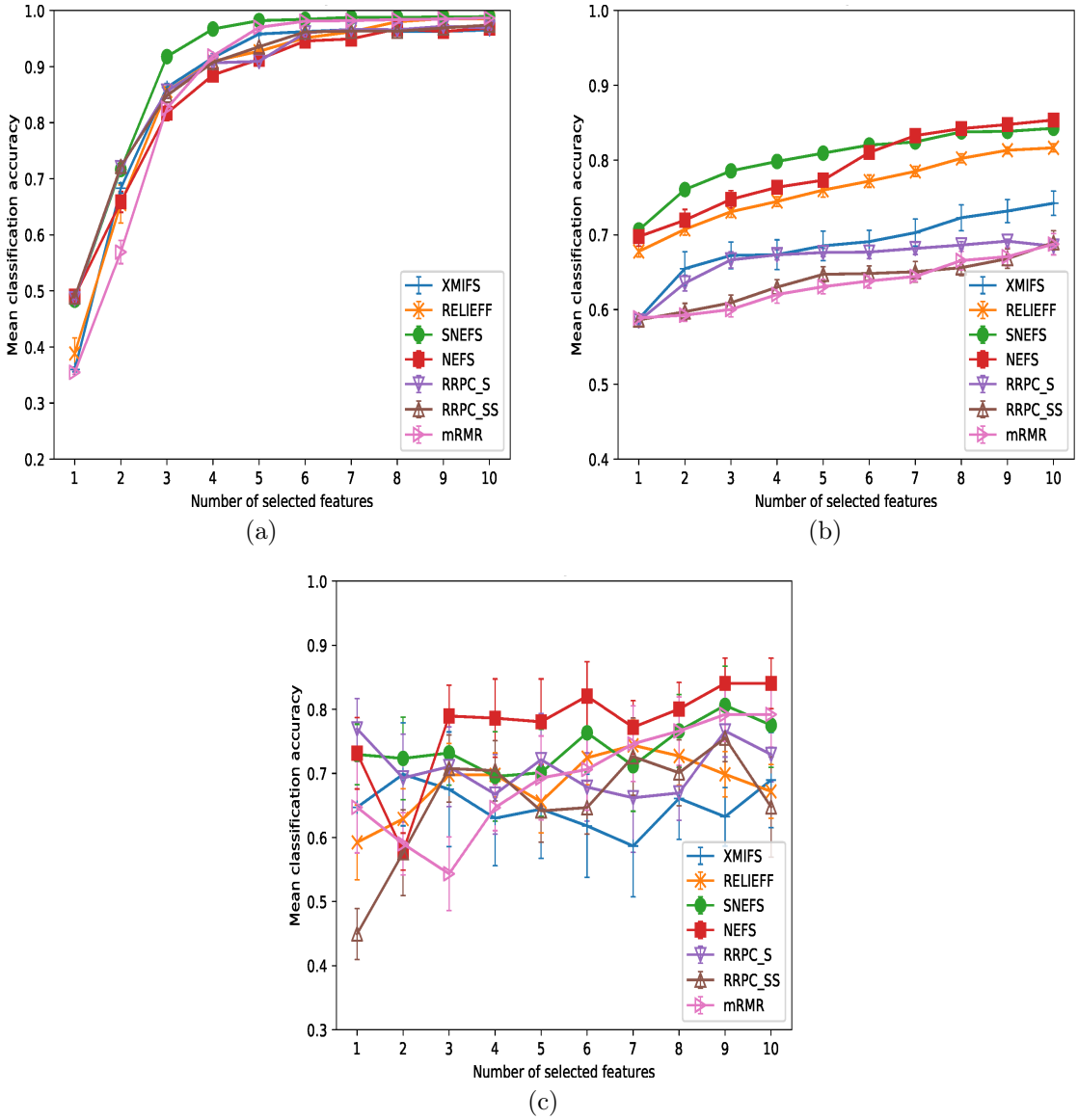
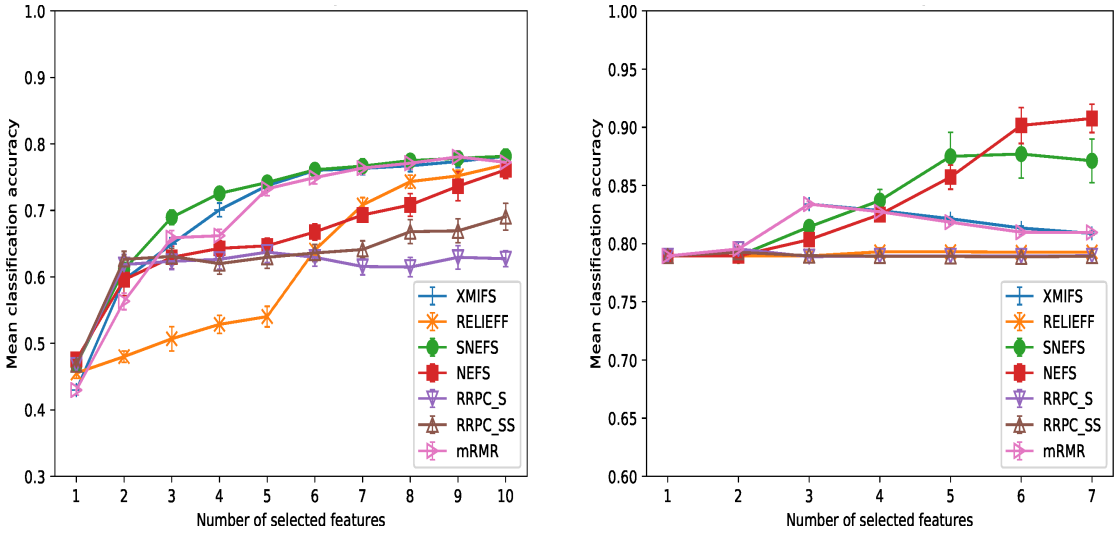


Figure 2.3: The classification accuracy for the MLP on GAS (a), and for the RF on SPAMBASE_N (b) and on a small random sample of SPAMBASE_N (c).

2.5.1 CPU Time Models

In this Section, we compare the algorithms in terms of the CPU time. We recall the complexity analysis of Section 2.4 to extend it to RELIEFF, X-MIFS and RRPC. When considering the asymptotic cost models of the different algorithms, we explicitly indicate the dependency on the number of selected



(a) (b)

Figure 2.4: The classification accuracy for the SVM on SEGMENT_N (a) and on HYPERSPHERES (b).

features s (whenever possible), since most of the experiments used $s \ll D$.

In [83], the authors show that the cost of RELIEFF for T iterations is:

$$\mathcal{C}_{\text{RELIEFF}} = O(DNT). \tag{2.13}$$

As concerns X-MIFS, the algorithm follows the same greedy procedure of NEFS. The main difference is the evaluation of the MI instead of the NE. The computation of the MI uses a modified version of (2.3), in which the probability distribution functions are replaced by the estimators based on multi-dimensional histograms. The implementation of the authors also includes the binarization of the features, so $|\mathcal{D}_{\mathcal{S}}| = 2^{|\mathcal{S}|}$. For each subset, X-MIFS estimates the joint and marginal probability distributions through a linear traversal on all the points. The summation in (2.3) is also linear in $2^{|\mathcal{S}|}N_c$. Therefore, the worst-case time complexity of X-MIFS for selecting s

Table 2.3: Asymptotic time complexities.

NEFS/SNEFS	RELIEFF	X-MIFS	RRPC_S/mRMR-MIFS
$O(DN^{\alpha}s^2)$	$O(DNT)$	$O(DNs^2)$	$O(DNs^2)$

features is:

$$\mathcal{C}_{\text{X-MIFS}} = O\left(\sum_{i=0}^{s-1}(D-i)[N(i+1)+2^{i+1}N_c]\right). \tag{2.14}$$

By using appropriate data structures to exploit the sparsity of the bins, and under the same hypotheses made in Section 2.4, the time spent on the traversal of the histograms can be reduced significantly, and the cost of X-MIFS becomes $O(Ds^2N)$.

As concerns RRPC, the correlation can be computed in linear time with respect to the number of samples. The worst case is the supervised scenario, in which the cost of computing the correlation is always $O(N)$. At each iteration, one needs $O(|\mathcal{S}|)$ operations to compute (2.7) and update the maximum. The complexity is then:

$$\mathcal{C}_{\text{RRPC\_S}} = O\left(\sum_{i=0}^{s-1}(D-i)iN\right) = O(Ds^2N). \tag{2.15}$$

By using similar arguments, it can also be shown that the complexity of mRMR-MIFS is equal to that of RRPC. The asymptotic costs of all the algorithms are summarized in Table 2.3.

As concerns the scalability of the different implementations, in Table 2.4 we report the least-square linear models of the running time with respect to the size of the dataset DN. We recall that s is fixed to 10, T to $N/4$ and ϵ to 15. It is evident that X-MIFS is the fastest. This can be explained if one considers that X-MIFS is applied on binary features and the current implementation is very efficient and uses bitwise operators. RELIEFF, RRPC and

Table 2.4: CPU Time Linear Models.

Method	Coefficient	Intercept
X-MIFS	4.8×10^{-7}	-0.144
RELIEFF	1.14×10^{-6}	1.21
SNEFS	1.02×10^{-5}	16.6
NEFS	2.61×10^{-4}	257
RRPC_S	3.8×10^{-6}	-0.521
RRPC_SS	3.2×10^{-6}	-0.489
mRMR-MIFS	1.57×10^{-6}	-0.152

mRMR-MIFS are also fast. NEFS is the slowest because of the cost paid for the construction of multiple indexes as well as the high frequency of nearest neighbors searches. RELIEFF instead updates the entire weights vector for each iteration, thus reducing the number of searches for the nearest neighbors. SNEFS is slower than most of the other methods but significantly faster than NEFS. Therefore, when robustness to noise and to class imbalance is a key requirement, SNEFS is a valid choice because it tends to provide better results in a reasonable amount of time.

Chapter 3

Learning with Ensembles

In contexts characterized by high dimensionality and data scarcity, feature selection is not always sufficient to learn accurate models. In particular, filter methods requiring large datasets for the correct estimation of relevance scores are usually not effective with a limited number of samples, because they can suffer from higher levels of noise and uncertainty. In this context, a possible approach to learn robust models is group or ensemble learning. In this Chapter, we focus on a case study, related to salary prediction in the IT job market, to show that ensembles represent a valid alternative to dimensionality reduction when the latter is not beneficial because of data scarcity or noise.

Section 3.1 introduces the application domain and some related work. Section 3.2 describes the followed procedure for collecting data as well as our feature engineering and data cleaning processes. Section 3.3 defines the classification models used for salary range prediction, including linear models, neural networks, random forests, boosting machines and voting ensembles. Section 3.4 reports the results of the model comparison in terms of classification accuracy, precision, recall and Area Under the ROC curve (AUC). Experiments show that additional feature selection does not lead to any significant advantage with respect to our initial feature engineering and data

cleansing strategies. In addition, we empirically prove that robust classifiers can be built by using voting ensembles, which lead to an accuracy of $\approx 84\%$.

3.1 E-Recruitment

Accurate recruitment of employees is a key element in the business strategy of every company because of its impact on productivity and competitiveness. Nowadays recruitment processes have evolved into complex tasks, which involve rigorous evaluations and interviews of candidates, with very high commitment requirements for both the companies and the candidates. Internet and web technologies can significantly help companies and job seekers in finding the best possible match, therefore e-recruitment has become an essential element of all hiring strategies. Several web portals have been developed, such as *CareerBuilder*, *Monster* and *Tecnoempleo*, and social networks for professionals like *LinkedIn* are becoming increasingly popular.
The impact of e-recruitment on the industry is a very active topic of research and has been addressed by several studies [68, 122, 85, 94, 33, 15]. Recently, a framework for candidate ranking and résumé summarization has been proposed to improve screening performance [116]. Other applications include a tool for candidate evaluation that adapts to the feedback received [51], an approach to automatically evaluate a CV [50], and a scoring system to filter candidates and reduce the workload of recruiters [112].
Several systems have also been proposed to recommend jobs to candidates according to a profile obtained by using clustering techniques [7, 69, 67, 4, 70, 117]. In [128], a model is proposed for detecting talent and updating the knowledge taxonomy of a company to help recruiters search for the professional profiles the company lacks. A hybrid approach is presented in [8], where job offers are grouped by using supervised machine learning combined with expert labeling. ML algorithms have also been employed to predict

the urgent need of a specific skill [1]. Focus has been put also on systems and databases enriched by data mined from web portals and social networks [75, 36]. Expert retrieval systems have been built from user profile information coupled with user behavior inside different social networks [28] as well as with location-based data and connections to potential candidates [142].

Most of the works that focus on the extraction of insights from e-recruitment portals retrieve the information associated with each job post as text and then they represent each sample as a vector of word/keyword frequencies. As a consequence, these vectors are often characterized by a very high number of dimensions (in the order of thousands). Therefore, it is necessary to collect huge amounts of job posts to be able to train a model effectively. However, for websites with a limited target audience, such as portals developed for a specific geographic area or job sector, there are relatively few job posts. Among these, only a small percentage has an explicit indication of the offered salary. Therefore, learning an accurate model for salary prediction can be a challenging task.

In this Chapter we present a case study on salary prediction based on data collected from an e-recruitment website specifically designed for IT jobs in Spain, called Tecnoempleo[1]. The website contains a large collection of job offers, containing many machine-readable fields which are not common in other similar sites, such as the requested skills. However, only a small portion of posts include the offered salary. As a result, our collected dataset, which covers a period of 5 months, includes $\approx$ 4000 job posts, which are represented as vectors of $\approx$ 2000 features. In the next Sections, we describe our approach for dimensionality reduction and data cleaning, and we propose techniques based on voting ensembles for the prediction of salary ranges. Experiments suggest that our model can be effectively employed by an e-recruitment website to provide an automatic categorization of job posts by

[1]http://www.tecnoempleo.com (last access: March 2019)

salary range, even when the real offered salary is missing, or be used as a building block in a job recommender system.

3.2 Feature Engineering and Data Cleaning

The dataset under study includes 3970 job posts. Samples were collected by using a Python-based web crawler[2], which was run on a daily basis from December 2015 to April 2016. The raw data collected by using the crawler cannot be used directly without an accurate preprocessing phase to remove the sources of noise and normalize the remaining data. In addition, one also has to decide how the missing values should be treated. In this case, samples with missing values were removed because in most cases they cannot be substituted by any default value. Duplicates were also removed because they do not provide useful information for model training.

Unstructured information, such as the title of the job post as well as the textual description of the requested profile and the offered position, can be useful for extracting relevant features by employing text-mining algorithms like TF/IDF. However, in our specific application, we noted that textual descriptions are not always available or present errors and high variability. To simplify preprocessing and reduce the complexity of the learned models, we do not use unstructured data, since an extensive set of structured features is already available after web scraping.

Among the technology keywords that describe each post, we also noticed that some of them refer to tags consisting only of numbers, which probably describe the versions of the software required by the companies. However, these numbers, without technology names, are clearly not useful for salary prediction. Therefore, we remove them from the dataset to reduce dimensionality and possible sources of noise and uncertainty.

[2]See BeautifulSoup library, available at http://www.crummy.com/software/BeautifulSoup/ last access: March 2019

Since most of the keywords are manually introduced by users, it is also essential to unify the data format by merging semantically equivalent words, their different translations (e.g. in English or Spanish) and typographic errors. For example, keywords like *administracion, adm, administrativo, administration* should appear in the dataset as a unique binary feature: *administration*. Part of this process might be automated by using dictionaries, but it becomes more difficult for error correction. A way to improve the categorization of the different job posts on the web portal is to give the user the possibility to choose only from a predefined set of keywords. In this way one also prevents the sparsity characterizing our dataset.

In our experiments, we do not consider all the keywords that appear less than 10 times (corresponding to 0.25% of the total number of job posts). This results in the removal of hundreds of features that can be considered as noise because they do not provide enough information content. We also remove posts for jobs not located in Spain, which can be the result of erroneous posting by automated services outside the geographical region of interest.

By removing missing values, noisy features and unstructured data, it was possible to reduce the number of features to ≈ 200 from ≈ 2000 in the raw dataset as collected by the crawler. This result would be difficult to achieve by using domain-agnostic feature selection.

An important role in the feature preprocessing has also been given to the translation of categorical features into numerical features. The feature describing the *dedication* (full-time, part-time or autonomous work) is transformed into the maximum number of week hours: 40 for full-time jobs and 20 for part-time and autonomous jobs. Autonomous jobs are considered equivalent to part-time jobs because most of the times they refer to limited periods of time. The feature that describes the *incentives* offered by a company is substituted by a binary feature that indicates the presence or absence of incentives. The minimum *education* level (described by words) is converted to

Table 3.1: Correspondence between education levels and minimum number of years in the education system for Spain.

Education level	Min. years in education
School	10
High School	12
Professional formation (PF)	12
Higher PF	14
Junior Engineer	15
Graduate (Bologne)	15
short Degree	15
Engineer	17
Degree	17
Msc. (Bologne)	17
PhD	20

the minimum number of education years according to the Spanish education system. All the levels that do not fall in the main categories as listed on the website are assigned the same number of years of compulsory education (10 years, age 16). For the other values, see Table 3.1.

For each of the 488 companies that posted a job offer, we retrieve the approximate number of employees. This is used as an indicator of the company size, which can be related to the number and frequency of job posts. For most of them we retrieve the information from their public social profiles (e.g. LinkedIn). In case this information is not publicly available, the information is completed with the median value of workers in the rest of the companies. The textual representation of working experience is translated into the average number of years. For example, *2years* is substituted by 2, while *3-5years* by 4. We describe the type of *contract* by using 3 mutually-exclusive binary features (represented with the so-called *one-hot* encoding),

which indicate whether a job post is related to a permanent, temporary or other type of position (e.g. a hourly service). We also introduce the *per capita* gross product in the geographic region of each company. This is reasonable if one considers that the highest salaries are usually clustered around the most important economic centers.

We also rescale all features to the interval $[0, 1]$ by normalization on the range given by the maximum and minimum values of each feature.

The main focus in this Chapter is on the prediction of salary ranges, because this can result in a better categorization of the job posts and thus an easier navigation for the end-users. This is also motivated by the specific case study we are addressing, that is, predicting the salary associated with a job post, when the number of posts with an explicit indication of salary is low. In this context, the prediction of discrete ranges should be more accurate than the prediction of the actual salary, which is a continuous number. To accomplish this, we translate the original dataset suitable for regression into a dataset useful for classification. Each job post is assigned to one of four classes, which represent salaries in the low, medium-low, medium-high and high ranges. These ranges are decided according to the values of the first, second and third quartiles (also known as Q1, Q2, and Q3).

3.3 Models for Salary Range Prediction

After formulating the prediction problem as a classification task, we compare different models for finding the classifier that best explains the data in a job recruiting scenario characterized by high level of noise, high dimensionality and a limited set of samples. We consider several solutions based on:

- Linear models (LM);
- Logistic regression (LR);

- K-nearest neighbors (KNN);
- Multi-layer perceptrons (MLP);
- Support vector machines (SVM);
- Random forests (RF);
- Adaptive boosting with decision trees (AB);
- Ensembles of the previous models.

LM, LR, KNN, MLP and SVM represent some of the most effective (individual) models based on numerical data, as confirmed by recent applications [48, 110, 32, 5, 106, 144, 135, 145]. However, in our real-world scenario, which is characterized by high dimensionality and uncertainty, there could be no clear winner among several individual classifiers. Combining two or more models into a *committee* or an *ensemble* can be beneficial with respect to the classification performance of the overall system and its robustness to noise. Consequently, we also evaluate the performance of random forests (RF), which employ a set of decision trees trained on random subsets of the original data, and boosting algorithms, in particular the well-known Adaptive Boosting algorithm (AdaBoost or AB), which is iteratively built upon one simple model that is progressively improved (or *boosted*) by penalizing samples misclassified in the previous iteration. Recent applications of RF and AB suggest that they can lead to more accurate models, especially in the presence of high levels of noise [107, 66, 118, 102].
As described in Chapter 1, other approaches for ensemble learning are possible such as voting classifiers. In this case a committee of heterogeneous weak models is trained so as to have several independent predictions of the class of a sample. The final decision on the output variable is then the result of a majority vote among the members of the committee. In our work we use

two variants of a voting classifier: one that includes all the other classifiers that we compare in the tests (*Vote*), and one that includes only the top-3 best performing models (*Vote3*). In this way, we also evaluate the effect of choosing large or small committees on the accuracy of the final classifier. Having larger ensembles does not always lead to superior results, especially when individual members are treated equally, and learners that are not able to extract the information content in a specific scenario have the same weight of the other learners in the ensemble. When the number of bad learners is high, the uncertainty in the final predictions can increase.

3.4 Experiments and Discussion

We use the implementation that is provided by the Python library *scikit-learn* [104]. In particular, we employ several *pipelines* consisting of 3 main stages:

1. Data normalization to the interval $[0, 1]$;
2. Feature selection;
3. Classification according to one specific model characterized by the best configuration of parameters as retrieved by applying a previous step of grid search.

To perform the feature selection, we use the X-MIFS algorithm [30], in its original implementation developed by the authors, and the novel approach presented in Chapter 2. We consider these automatic feature selection methods to test for the existence of a subset of features that can improve the accuracy but would be difficult to extract during the manual preprocessing. Experiments show that in our application context, characterized by high dimensionality and data scarcity, an additional step of feature selection does

not give any advantage with respect to the steps of data cleaning and data augmentation described in Section 3.2.

3.4.1 Model configuration and selection

As concerns the (generalized) linear models (GLM), we compare a support vector machine with a linear kernel and LR. The LR models are trained with ℓ_1 and ℓ_2 regularization with 10 different weights chosen uniformly in the range $[10^{-3}, 10^3]$. The same regularization scheme is also employed for the SVM, with a maximum number of iterations fixed to 5000 and a convergence tolerance of 10^{-3}.

For the KNN models, we compare *exact* algorithms (i.e. with no approximated technique) based on the Manhattan ℓ_1-norm distance as well as the Euclidean ℓ_2-norm distance. We use two strategies to assign a class to a sample: one that assigns a uniform weight to each neighbour and one that weights the contribution of a neighbour according to the inverse of its distance from the sample so as to give more importance to the closest points. k varied within the set $\{1, 2, 4, 8, 16, 32\}$.

The method used for the optimization of the MLP weights is *stochastic gradient descent* (SGD). To reduce convergence time an adaptive learning rate is used, with the addition of early stopping to avoid overfitting. Starting from an initial value of 1, the learning rate is kept constant as long as the training loss keeps decreasing. Each time two consecutive epochs fail to decrease the training loss or to increase the classification accuracy on the validation set by a tolerance value (10^{-3}), the current learning rate is divided by 5. The training stops after a maximum number of epochs (5000) or when the classification accuracy on the validation set does not increase after 50 epochs. The network used for the tests is shallow, with only one hidden layer of neurons and with the *tanh* activation function. We compare different models with a number of neurons varying in the set $\{1, 2, 4, 8, 16, 32\}$ and

with 10 ℓ_2-regularization weights equally spaced in the interval $[10^{-3}, 10^3]$.

As concerns (non-linear) SVMs, we evaluate models based on radial basis function as well as sigmoid kernels, with a kernel coefficient varying in $[10^{-3}, 10^3]$ and 10 different penalty parameters of the error term taken from the interval $[10^{-3}, 10^3]$. Even in this case the maximum number of iterations is fixed to 5000 with a tolerance for the stopping criterion equal to 10^{-3}.

Experiments include also RF classifiers trained by using the *bootstrap* technique. Each tree is trained by using either the Gini impurity criterion or the Information Gain criterion. The number of trees varied in the set $\{1, 2, 4, 8, 16, 32\}$.

The same criteria and number of trees are also employed in the case of the AB classifier. For Vote and Vote3 we use the average predicted probabilities (*soft voting*) to predict the classes.

For each configuration of the different models, we also investigate the effect of selecting 10 or 20 features (as opposed to considering all the features) by using X-MIFS or NEFS.

In order to select the best configuration for each model, we perform a grid search on 90% of the data by using a 3-fold cross validation and by selecting the configuration with the best average classification accuracy. In Table 3.2 we report the optimal parameters for the different classifiers (excluding the voting classifiers). The use of feature selection (with NEFS as well as X-MIFS) is not beneficial. This empirically proves that our customized feature preprocessing already removes (almost) all the possible sources of noise and redundancy through the procedure described in Section 3.2.

As concerns Vote and Vote3, we do not consider the best configurations of the models individually but we train all the models from scratch and validate the voting classifiers independently from what are the best configurations for GLM, KNN etc. The reason behind this is that the performance of a voting ensemble does not always improve as the performance of the voting members

Table 3.2: Optimal number of features (FS) and model configurations for all the classifiers (excluding Vote and Vote3).

Pipeline	FS	Model hyper-parameters
GLM	all	model: LR, regularization: ℓ_2, reg. weight: 0.009
KNN	all	distance: ℓ_1-norm, k: 8, neighbor weight: distance inverse
MLP	all	hidden layer size: 8, reg. weight: 0.001
SVM	all	kernel: RBF, kernel coeff.: 111, reg. weight: 0.009
RF	all	split criterion: Gini, # of trees: 32
AB	all	split criterion: Gini, # of trees: 32

improves. Sometimes weakening one member to decrease its importance in the vote can be beneficial for the final decision. In other cases the difference between two configurations for a single classifier is so negligible that the final vote is not affected at all. This is confirmed by our experiments, with Vote and Vote3 obtaining better accuracy with configurations that are different from those in Table 3.2. The optimal hyper-parameters for the voting classifiers are reported in Table 3.3. Each model resulting from the grid search is then trained and evaluated on the entire dataset by using 10-fold cross validation.

3.4.2 Model comparison

The scores used for comparing the different algorithms include:

- the classification *accuracy*, which is the percentage of correctly classified samples;
- the *precision*, which is defined, for one specific class, as the fraction of true positives among the samples predicted to belong to that class (true and false positives);
- the *recall*, which is defined, for one specific class, as the fraction of true

Table 3.3: Optimal configurations for Vote and Vote3 (differences from Table 3.2 in bold).

Vote	
Model	Hyper-parameters
GLM	model: LR, regularization: ℓ_2, reg. weight: **0.0045**
KNN	distance: ℓ_1-norm, k: **16**, neighbor weight: distance inverse
MLP	hidden layer size: 8, reg. weight: 0.001
SVM	kernel: RBF, kernel coeff.: 111, reg. weight: 0.009
RF	split criterion: Gini, # of trees: 32
AB	split criterion: Gini, # of trees: 32

Vote3	
Model	Hyper-parameters
KNN	distance: ℓ_1-norm, k: **16**, neighbor weight: distance inverse
RF	split criterion: **InfoGain**, # of trees: 32
AB	split criterion: Gini, # of trees: 32

positives among the samples belonging to that class (true positives and false negatives);

- the F_1 score, which is the harmonic mean of precision and recall;
- the area under the precision-recall curve (*AUC-PR*) built for different thresholds of the probability of the positive class;
- the area under the ROC curve (*AUC-ROC*) that shows the true positive rate against the false positive rate for different thresholds of the probability of the positive class.

For all the class-specific scores, we compute the average on all the classes in order to obtain a scalar value for each model. Table 3.4 summarizes the results for all the classifiers in terms of average accuracy, F_1 score, AUC-PR and AUC-ROC, with the indication of the corresponding standard errors.

Table 3.4: Average scores and standard errors for all the classifiers (best scores in bold).

	Accuracy	F_1 score	AUC-PR	AUC-ROC
LR	0.586 ± 0.0077	0.569 ± 0.0085	0.603 ± 0.0075	0.806 ± 0.0051
kNN	0.792 ± 0.0067	0.792 ± 0.0066	0.891 ± 0.0055	0.938 ± 0.0034
MLP	0.591 ± 0.0150	0.552 ± 0.0230	0.659 ± 0.0110	0.831 ± 0.0077
SVM	0.663 ± 0.0066	0.669 ± 0.0076	0.881 ± 0.0062	0.930 ± 0.0049
AB	$\mathbf{0.836 \pm 0.0069}$	$\mathbf{0.837 \pm 0.0068}$	0.883 ± 0.0060	0.936 ± 0.0036
RF	$\mathbf{0.840 \pm 0.0076}$	$\mathbf{0.838 \pm 0.0081}$	0.905 ± 0.0055	0.949 ± 0.0028
Vote	$\mathbf{0.844 \pm 0.0028}$	$\mathbf{0.843 \pm 0.0028}$	$\mathbf{0.917 \pm 0.0036}$	0.960 ± 0.0018
Vote3	$\mathbf{0.837 \pm 0.0073}$	$\mathbf{0.837 \pm 0.0073}$	$\mathbf{0.923 \pm 0.0034}$	$\mathbf{0.963 \pm 0.0017}$

Furthermore, Figures 3.1-3.4 provide box plots as well as PR and ROC curves. The classifiers based on ensembles of decision trees (AB and RF) as well as those based on voting ensembles (Vote and Vote3) achieve the best accuracy. Their average accuracy is ≈ 0.84, with the voting classifiers that lead to a slightly better median accuracy (≈ 0.841 for Vote and ≈ 0.85 for Vote3).

It is also evident that Vote is the most robust. This confirms that in our context a larger committee tends to reduce the variance of the final decisions, while a smaller ensemble based on the best learners leads to higher uncertainty. KNN achieves an average accuracy of ≈ 0.79 and sometimes is comparable to AB or RF. All the remaining models (LR, MLP and SVM) behave significantly worse. For LR, this can be explained by the evident non-linearity of the problem, while for MLP and SVM the scarcity of the data probably represents the biggest obstacle. In the case of the MLP, the best configuration selected 8 neurons and not bigger values such as 16 or 32, thus suggesting that the size of the network is not relevant for the classification. The limited set of samples and the categorical nature of most of the features make the MLP and SVM models ineffective.

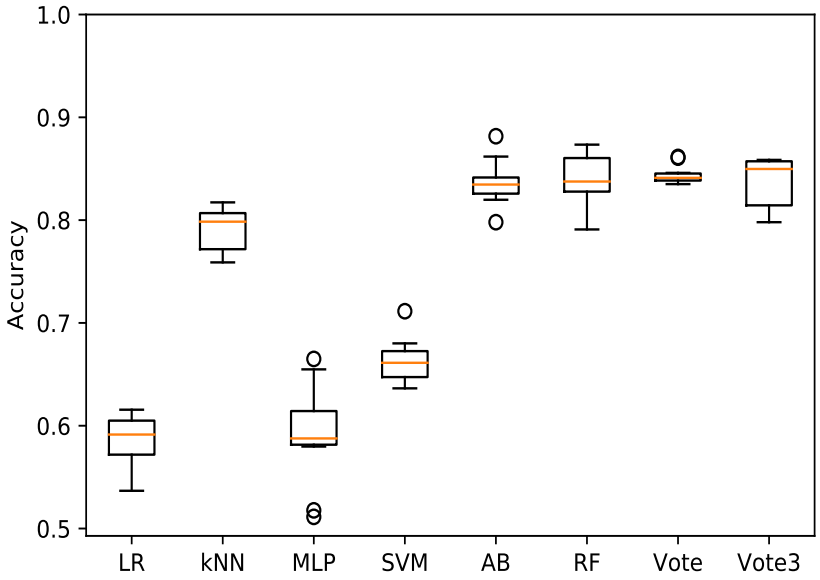


Figure 3.1: Box plot of classification accuracy.

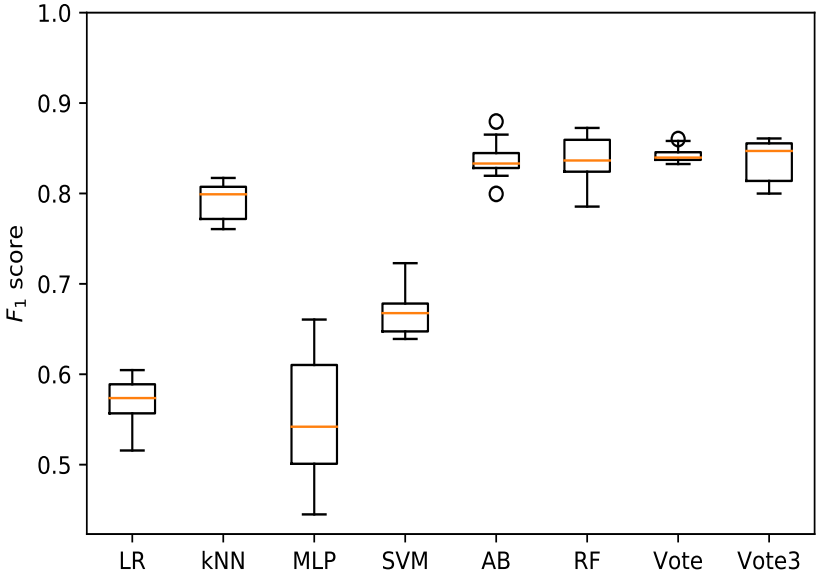


Figure 3.2: Box plot of F_1 score.

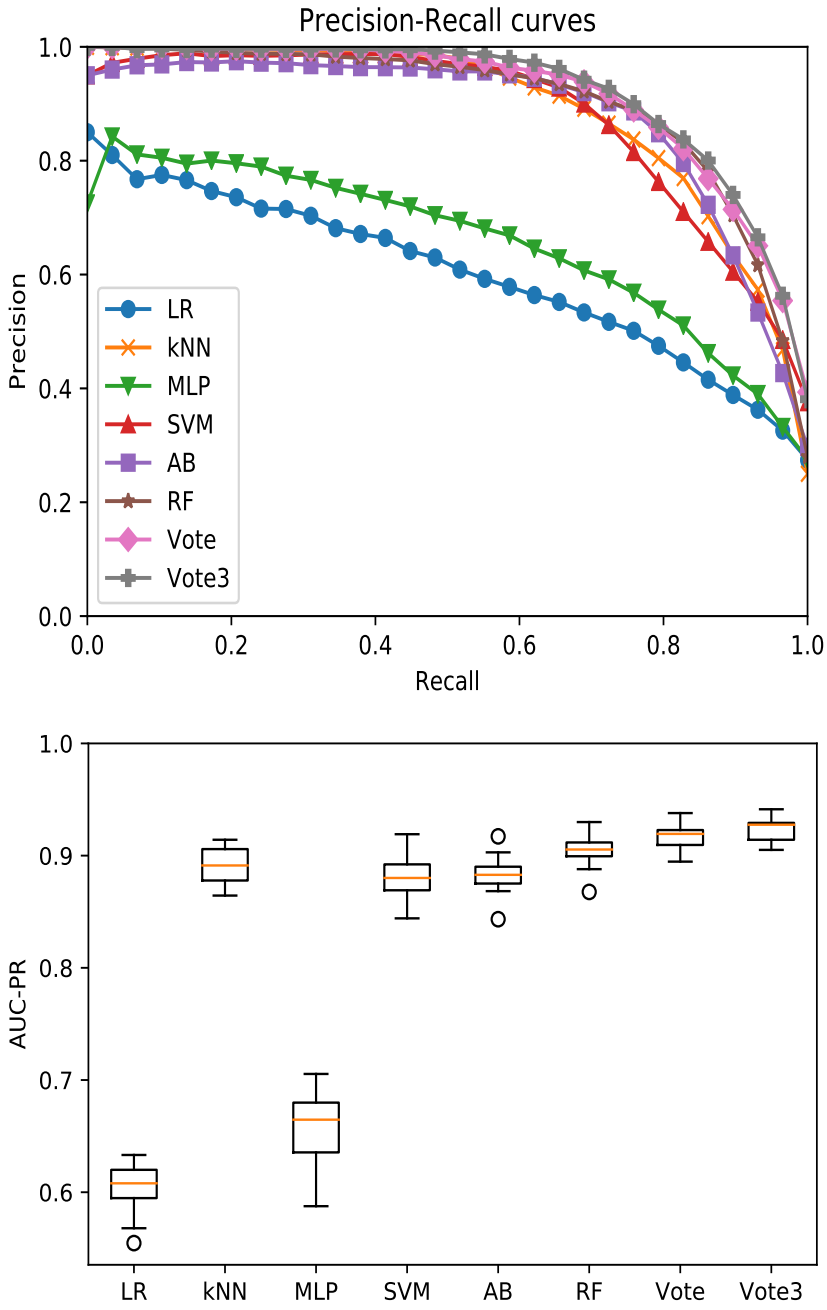


Figure 3.3: Precision-Recall curves with corresponding AUCs.

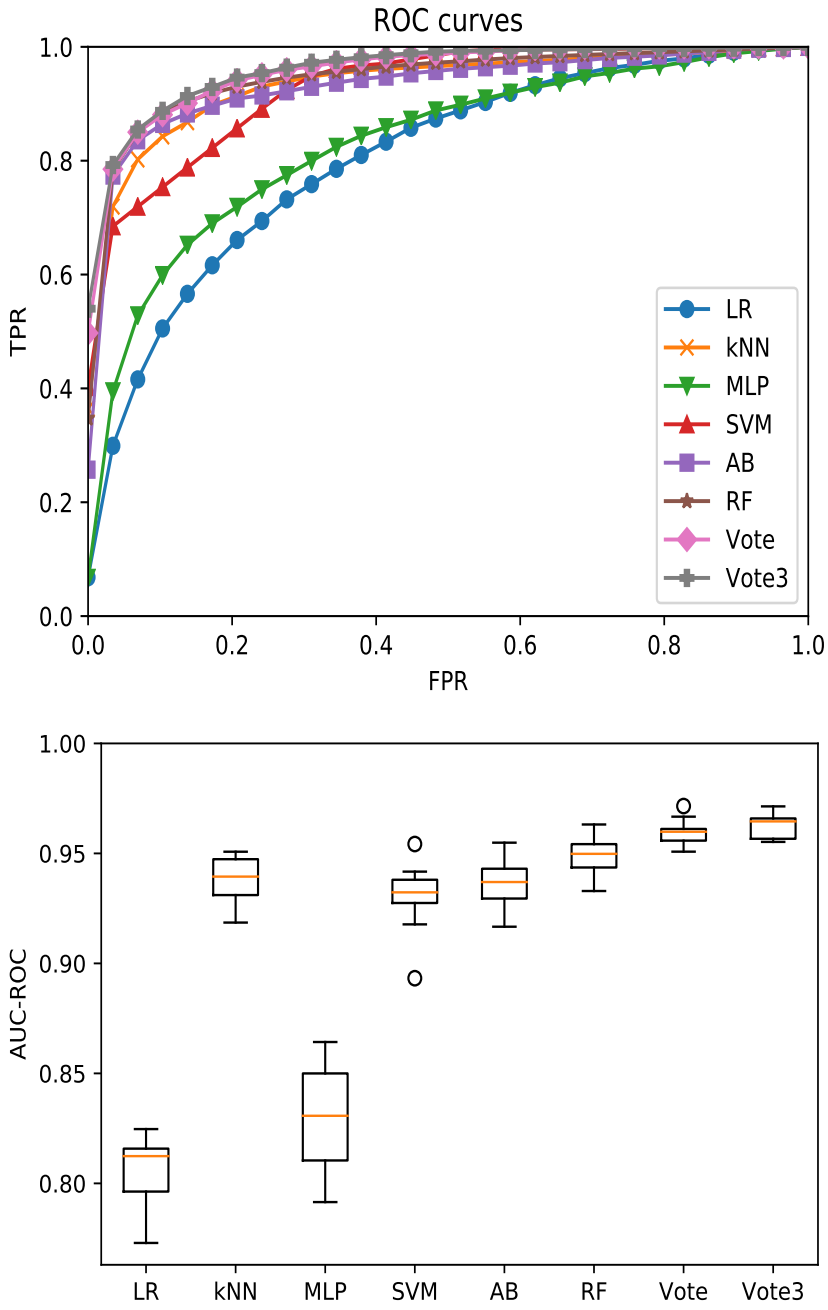


Figure 3.4: ROC curves with corresponding AUCs.

This explains also why the classifiers that can easily deal with categorical features as those based on decision trees (AB and RF) behave generally better. These findings are also confirmed by comparing the models on the F_1 score, which accounts for the precision and the recall measures simultaneously. The best average score (≈ 0.843) is achieved by Vote, with RF the second-best (≈ 0.838). The worst results belong to the MLP, which is also the model with the largest variance.

The previous scores describe the performance of each model in recognizing all the classes. In order to get more insight into the capabilities of each classifier, the experiments that led to the results reported in Figures 3.3 and 3.4 focus on a slightly different setting. The same configurations used in the previous tests are employed in a *one-vs-rest* scenario, in which we compute the precision, recall, true positive rate and false positive rate in the context of a binary classification. For each class, we set to 1 all the members of that class and set to zero all the remaining samples. We repeat this for all the classes and at the end we average the results obtained. This new setting is interesting because it can describe the behavior of the classifiers with unbalanced classes. It is possible that a model that achieves poor results in the original context is actually better in recognizing one specific class as opposed to all the others.

The best performing models of the previous set of tests continue to achieve the best results also in the new context. However, the SVM performance improves significantly and now is comparable to KNN, AB, RF, Vote and Vote3. This is due to the simplification induced by the new scenario, which is probably characterized by a separating surface that is easier to learn. However, this surface is still non-linear, as it is evident by looking at the poor performance of the LR. The MLP continues to perform poorly even in the new simplified scenario, thus suggesting that the model is not able to extract useful information from the data because of the limited amount of samples.

Perhaps deeper architectures are necessary to find more informative features but these models will probably be affected by the scarcity of the data even more. Vote and Vote3 consistently lead to superior results, thus showing that, in contexts characterized by high dimensionality, uncertainty and limited samples, ensemble learning can effectively reduce the variance of the final predictions and lead to more robust models.

Chapter 4

Learning from Aggregated Data

In the extreme case when one has no access to individual samples, traditional instance-based ML algorithms cannot be used, including the feature selection and ensemble methods presented in the previous Chapters. However, we conjecture that, under the hypothesis of having access at least to aggregated data, the extraction of useful insights and guidelines is still possible by simulating parametric and stochastic models based on the available information. Especially for real-world phenomena that are difficult to sense and measure, the level of uncertainty can be very high and tend to infinity when no sample is available. Nonetheless, in real scenarios one can exploit expert knowledge to build models that can be used as a baseline or bootstrap sample to infer interesting patterns. This resembles Bayesian inference, which allows one to learn models that incorporate the prior knowledge about a problem.

To support our claim, in this Chapter we present a case study in hotel revenue management, and in particular we address the problem of finding an optimal pricing policy for hotel rooms to maximize revenue. Section 4.1 introduces the application domain, with a brief overview of some state-of-the-art solutions for dynamic pricing in hotel revenue management. Section 4.2 describes in more detail the hotel reservation process and our proposed simulator of room demand. Section 4.3 focuses on the definition of our parametric

and stochastic models of reservations and cancellations, while Section 4.4 describes the chosen pricing policy and acceptance probability model employed in our experiments. Section 4.5 reports our results on aggregated data from 10 hotels in Trento, Italy. Experiments show that our simulator leads to results statistically consistent to the aggregated data used as input. In addition, we show that the adoption of optimized pricing policies based on our parametric and stochastic models leads to an average revenue increase of $\approx 19\%$ with respect to policies with fixed prices, and to low or absent risk of losses for small and medium/big hotels, respectively.

4.1 Dynamic Pricing in Hotel Revenue Management

Information Technology drastically changed how people plan and manage travels. The Internet revolutionized the way travelers can get information about trips and hotels. Tools such as search engines, online travel agencies or price comparison websites are now used for travel planning by people of different generations [134]. As a consequence, potential tourists can take decisions which are much more informed than before, and companies need to be competitive in order to survive. In fact, many organizations in the travel and tourism industries use IT to propose products online, thus reducing commissions that would be given to intermediaries [31]. Proposing the right price to potential customers is a very important aspect, with a direct impact on the revenue of companies. In particular, promising results have been achieved for airline ticket reservations [134]. However, other processes like hotel reservations or car rentals have not been studied extensively. Most of the research on hotel revenue management has focused on the transfer into the hospitality realm of the techniques used for airlines, and frequently only general results on dynamic network management have been shown [23].

Despite some evident similarities, hotel reservations present significant dif-

ferences from airline reservations. In the case of a flight ticket, the customer can book a direct flight or multiple connecting flights. When one flight is canceled, usually the remaining connections in the ticket are also canceled. In the case of hotels, a similar situation can happen when a customer books rooms in different hotels for consecutive stays, but this is not a common use case, especially if we consider that recently the number of nights in hotel reservations has been decreasing as an effect of economic instability. Another major difference between airline tickets and hotel reservations is the duration of the service offered to the customer. For a flight, the duration is not decided by the passenger but by the airline company together with other governance institutions. For a hotel room, the length of stay is chosen by the customer, which is free either to adapt to the hotel room availability in case the original request cannot be satisfied or to search for another hotel. In particular, finding alternative hotels is often easier than finding a different airline covering the same connection. The differences between the two domains explain the different approaches in the literature and the increasing attention in developing customized solutions for the specific problems related to hotel revenue management.

Within the hospitality realm, research on revenue management (RM) has considered diverse possibilities for optimizing revenue, from marketing to price control. Optimization problems related to RM are usually expressed following two main approaches [120]: capacity control [25, 13, 87, 62] and dynamic pricing [20, 11].
In capacity control, the decision variable is the number of offered rooms (including service levels). In contrast, in dynamic pricing the decision variable is price. Recent works mainly focus on dynamic pricing, to propose different price offers for a limited set of products with fixed capacities. In this Chapter, we address the problem of learning patterns and guidelines for the maximization of revenue based on dynamic pricing. We focus on this process

because of its similarity to what happens on online booking platforms, where the same room can be sold at different prices according to factors like the length of stay, the time to arrival, the season and the occupancy level.
Dynamic pricing problems have been solved using various strategies [73, 42], including rule-based frameworks [37], linear programming [92], integer and dynamic programming [23].
The demand is considered to be deterministic [81] or stochastic [23]. Moreover, many techniques adopt the simplifying assumption that the demand is independent from the chosen policy. More complex scenarios, where demand can be influenced by other factors (e.g. price), are more difficult to handle and closed-form solutions are rarely available [24]. In addition, if sold rooms become available again before they are consumed by the customer (e.g. via stochastic cancellations), the dimension of the optimization problem grows exponentially and approaches like dynamic programming are effective only in specific cases [97]. A possible solution to mitigate the complexity of the model is given by approaches based on approximated dynamic programming [23, 141]. To reduce the computational cost, a linear relaxation of the original problem is considered. However, such approximated models are not flexible enough to include stochastic cancellations interspersed with reservations.

If problems related to hotel RM scenarios are computationally too complex to be solved exactly, the approximate maximization of revenue can be achieved by using simulation-based optimization [22, 53]. In this way, the analytical model can be substituted with a simulator of many inter-related processes like reservations, cancellations, no-shows and walk-in customers. To define simulators of complex systems like a hotel booking system, an effective technique is Monte Carlo simulation [119]. Generated events such as reservations and cancellations lead to a distribution of possible revenues. The expected value of that distribution is then considered as the variable to be maximized.

In [138, 20], a Monte Carlo approach is employed to simulate the demand. New reservations are forecast according to the distribution learned from history and by means of additive/multiplicative pick-up [139] or exponential smoothing [57, 9]. A similar approach can be used also with other forecasting models, such as regression or moving average models [131].
In [20], the effect of the price on the demand is also considered: lower and higher prices are associated with higher and lower acceptance probabilities, respectively. The pricing policy is modeled by a set of multipliers which can increase or decrease the price of a simulated reservation with respect to the average price learned from data. The parameters of the pricing model are optimized by using CMA-ES [64, 65].

However, the aforementioned Monte Carlo method is based on historical records, and it does not provide a simple way for the user to run *what-if* analyses. In real scenarios, it is very common that the distributions characterizing the demand change in response to mutated environment conditions. For example, it is highly probable that the pricing policy of a hotel should change when massive events like concerts, exhibitions or sport events happen nearby. The hotel manager would benefit from an automated system that exploits historical data whenever possible, and at the same time allows to readily inject new valuable information as soon as it becomes available in order to test different scenarios.
Another limitation of [138, 20] is that cancellations are modeled as a set of events that are independent from reservation requests. In fact, cancellations are realized before the generation of new reservations. Consequently, the state of the hotel registry (e.g. the room availability) does not change dynamically after each event. The first-generated reservation would see an availability higher than the one that would be provided if all events are interspersed, as in a real booking scenario. Moreover, one cannot simulate the case of a customer booking a room and then canceling later during the same

day (or even immediately after).

We propose a system that simulates the hotel booking process by considering interspersed reservations and cancellations, and that defines arrivals using parametric and stochastic models that do not require individual historical records but only aggregated data. Our approach exploits expert knowledge and previous research results to develop models able to describe different realistic scenarios. We allow straightforward *what-if* analyses, while taking into account the inherent stochastic fluctuations of the processes involved. These models are then trained by using simulation-based optimization, as described in the next Section, to provide useful insights and statistically powerful guidance when historical data are not available or difficult to retrieve.

4.2 Simulation of a Hotel Booking Scenario

In hotel revenue management, it can be difficult to model all the inter-related processes that are present in a real scenario. Therefore, most research has focused on simplified and more tractable views of the world. If one is not bound to use models that provide an exact solution as with the traditional dynamic programming approach, more realistic and fine-grained models can be defined by using simulation-based optimization. In this Section, we define the main concepts related to hotel reservations, and then we provide an overview of our simulator and a description of the simulation-based optimization procedure that we use for learning.

4.2.1 Definitions

Let us now fix the notation and define the main concepts, before the general description of the system in Section 4.2.2.

Definition 4.1. A *reservation request* (RR) is an event characterized by the following features:

- the *reservation day* (RR_{res}), which is the day the request occurs;
- the *arrival day* (RR_{arr}), which is the day the customer arrives at the hotel;
- the *length of stay* (RR_{los}), which is the number of nights reserved;
- the *size* (RR_{size}), which is the number of rooms reserved.

Definition 4.2. A *reservation offer* (RO) is an admissible reservation request (for which there is room availability) characterized by the *price* (RO_{price}) proposed by the hotel.

RO_{price} depends on several factors, including the features of RR, the time a request arrives, the presence of extra services or the price proposed by competitors.

Definition 4.3. An *accepted reservation* or simply *reservation* (R) is a reservation offer accepted by the customer.

An accepted reservation is registered on the hotel registry, thus effectively changing the room availability.

Definition 4.4. The *acceptance probability* of a reservation offer ($\Pr_{\text{accept}}(RO)$) is the probability that a customer accepts RO and the proposed price, and therefore is equal to the probability that RO is registered on the book.

Definition 4.5. The *state of the hotel* $S(t)$ is defined as the state of the booking registry at time t, which corresponds to the historical records up to t (when available) as well as the set of reservations (for future arrival days) that appear in the registry at time t.

Before introducing formally the concept of booking horizon, we need to define the concepts of distance between two days and of time-to-arrival.

Definition 4.6. Given two days identified by $i, j \in \{0, 1, 2, \dots\}$, the number of days between i and j, or their *distance*, is:

$$d(i, j) = d(j, i) = |i - j| \geq 0.$$

Definition 4.7. Given a reservation R, the *time-to-arrival* of R is:

$$R_{\text{TTA}} = d(R_{\text{res}}, R_{\text{arr}}).$$

A similar definition is also valid for reservation requests and offers. $R_{\text{TTA}} = 0$ can represent two possible events. One is the event of a customer that books one or more rooms for one or more nights starting from the night of the same day of the reservation. The other is the event of a customer that arrives at the hotel with no reservation and asks for one or more rooms. In the literature, the customer associated with the second event is usually called a *walk-in* user. In this Chapter, we do not distinguish the customers associated with the previous events, and we refer to them as *walk-in* users.

Definition 4.8. The *booking horizon* (BH) is the maximum time-to-arrival allowed by the hotel.

To properly model a hotel booking scenario, it is essential to include the management of cancellations, which are defined as follows.

Definition 4.9. A *cancellation* (C) is an event characterized by the following features:

- the *cancellation day* (C_{day}), which is the day the event occurs;
- the *reservation* (C_{res}), which is the reservation on the book that is canceled by the customer.

When a reservation is canceled, it is removed from the hotel registry, and the associated rooms can be booked by other customers.

Definition 4.10. The *cancellation probability*, t days before arrival of a reservation R ($\Pr_{\text{cancel}}(R,t)$), is the probability that the customer associated with R cancels it exactly t days before arrival, with $t \in [0, R_{\text{TTA}}]$.

According to the previous definition, the probability that R is canceled within its lifetime is

$$\Pr_{\text{cancel}}(R) = \sum_{t \in [0, R_{\text{TTA}}]} \Pr_{\text{cancel}}(R,t). \tag{4.1}$$

Our simulator also allows one to consider opening and closing periods, and to simulate reservations accordingly.

Definition 4.11. The *reservation requests horizon* (RH) is the set of all the reservation days to be simulated. It corresponds to the values that each R_{res} can assume during the simulation.

Definition 4.12. The *arrivals horizon* (AH) is the set of all possible arrival days. It corresponds to the values that each R_{arr} can assume during the simulation.

Definition 4.13. The *optimization horizon* (OH) is the set of arrival days for which there is the need of an optimal dynamic pricing policy to maximize revenue.

Most hotel managers are interested in maximizing the total profit and not the revenue. In our experiments, we do not consider costs (fixed and variable), and therefore we maximize only the revenue. The simulator can be extended to include costs, but this is out of the scope of this Chapter, which focuses on methods to learn useful patterns from aggregated data and to reduce the high level of uncertainty in the stochastic phenomena involved.

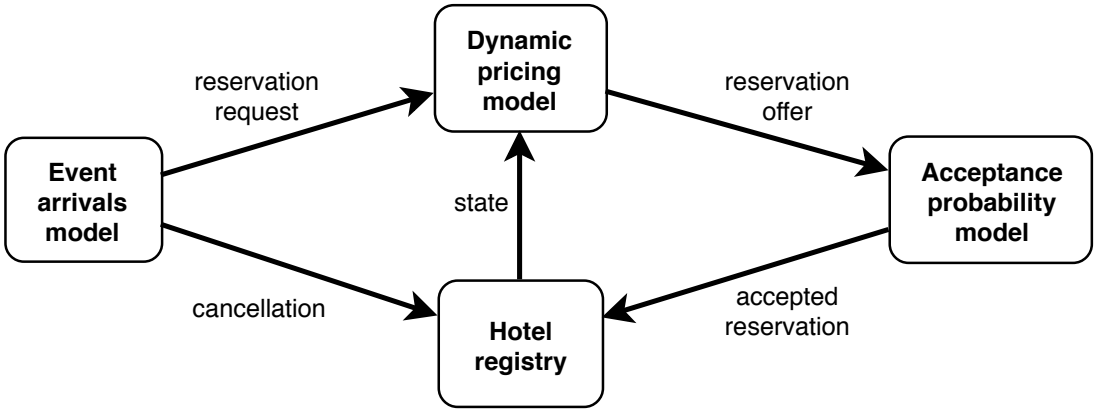


Figure 4.1: Dependency graph of the hotel registry and of the models used by our simulator.

AH represents the opening period of the hotel, and reservation requests that include days outside its bounds are rejected. RH and AH can overlap (entirely or partly) or be disjoint. OH is instead required to be a subset or to be equal to AH. The total revenue that drives the optimization procedure is evaluated on OH only. This leads to an increase in complexity but also to more flexibility, since one is able to simulate and compare different scenarios, including different opening periods and booking horizons.
The main models and components defined above are illustrated in Figure 4.1.

4.2.2 System overview

Differently from most traditional ML approaches based on a static dataset, we develop several components that simulate the hotel booking process, as described in the previous Section, and that generate data and optimize the pricing policy dynamically.
As depicted in Figure 4.2, our system includes:

- an event generator, which simulates reservation requests created by the customers as well as cancellations;

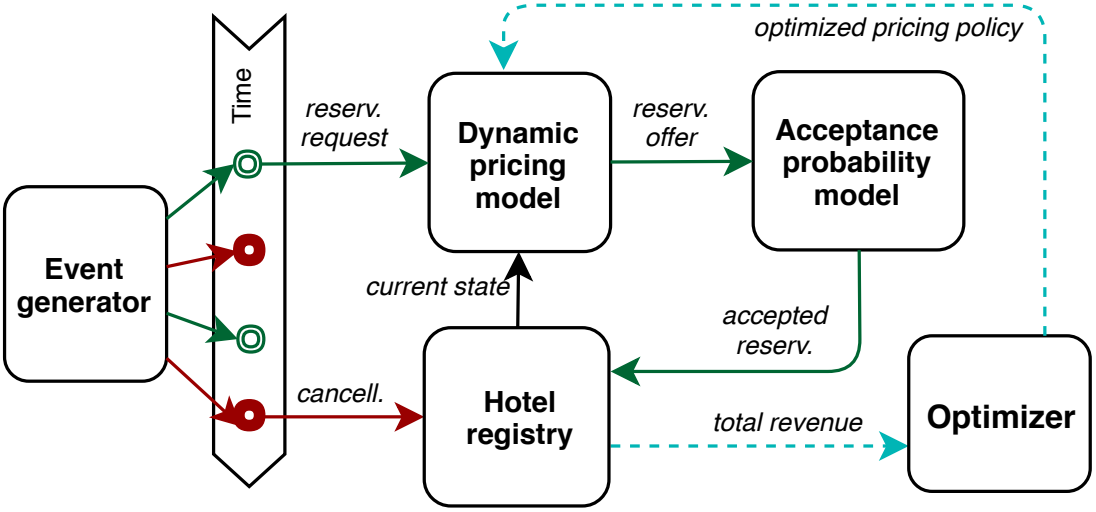


Figure 4.2: System overview. Reservation requests and cancellations are interspersed. The state of the hotel after one complete simulation is used by the optimizer to compute the total revenue and adjust the pricing policy.

- a registry, which stores the information about the state of the hotel, in particular accepted reservations and room availability;
- a dynamic pricing model, which proposes an offer for each reservation request;
- an acceptance probability model, which simulates the stochastic process by which customers accept or discard reservation offers;
- an optimizer, which searches for the optimal pricing policy to maximize revenue.

The event generator can also be preceded by a forecasting module. The models of future events can be learned from historical data by using techniques like the pick-up or exponential smoothing. However, we do not consider forecasting explicitly. Results do not change if we assume that the parameters of our models have already been predicted by a forecasting model.

In our implementation, for each simulated reservation day $r \in \mathrm{RH}$, a random sequence of $\mathcal{C}_r$ cancellations and $\mathcal{R}_r$ reservation requests is generated.

Each reservation request is associated with an arrival day $a \in \mathrm{AH}$ following or coinciding to r ($a \succeq r$). Each cancellation is instead associated with a specific registered reservation.
The proposal of a price depends on a reservation request and on the state of the hotel at the moment the event occurs. Since reservation requests and cancellations are interspersed, our procedure covers also the case of a customer reserving a room and, later on the same day, deciding to cancel it, maybe because of an error or because a better offer has been found.

Once a price has been proposed to the customer, a reservation is accepted according to the acceptance probability model. Then it is registered into the hotel registry and, if a cancellation does not occur until the end of the simulation, it is considered in the evaluation of the total revenue to be passed to the optimizer. As concerns the optimization, one objective function evaluation corresponds to the average total revenue of several simulation runs, with respect to the reservations recorded in the registry within the OH.

4.3 Parametric Models

One of the few examples of simulation-based approaches for hotel reservation dynamic pricing is the simulator in [138], which does not make any assumption on the distribution of reservation requests and cancellations. Reservation and cancellation models are learned completely from data. The hotel manager can only provide information about seasonality, in terms of duration and average prices. Since we are interested in defining models that can be effectively used by hotel managers even when historical data are not available or difficult to retrieve, we propose a parametric approach, by which the knowledge of domain experts can be exploited by using only a limited set of synthetic indicators. While it is almost impossible for the hotel manager to provide the entire reservation model without reverting to automated so-

lutions relying on history, it is a more viable option to ask for a synthetic indicator like the average number of reservation requests arriving in a certain period. Interviewed hotel managers confirmed that they often have access to aggregated information that can be exploited to maximize revenue.
In the following Sections, we propose a set of parametric models that can be used starting from the expected number of reservations and cancellations, expected length of stay and number of rooms. In addition, the proposed models are flexible enough to cover several, and sometimes very different, scenarios.

4.3.1 Simulation of reservation requests

Let $\mathcal{R}^a_r$, $r \in \text{RH}, a \in \text{AH}$, be the number of reservation requests generated on day r that are associated with arrival day a. The total number of requests generated within RH and associated with one arrival day is given by:

$$\mathcal{R}^a = \sum_{\substack{r \in \text{RH} \\ r \preceq a}} \mathcal{R}^a_r, \tag{4.2}$$

where $\preceq$ describes the relation *precedes or coincides to*.
The expected total number of reservation requests associated with one arrival day can be seen as the result of several independent processes, which occur on each simulated day within the BH of an arrival day:

$$\mathbb{E}[\mathcal{R}^a] = \Lambda(a) = \sum_{i=0}^{\text{BH}} \lambda(i, a), \tag{4.3}$$

where $\lambda(i, a)$ is the expected number of reservation requests occurring i days before the arrival day a.
If historical data are available, one can estimate directly $\lambda(i, a)$ for each i and a.

In our context, we define each $\lambda(i, a)$ by the following parametric model:

$$\begin{aligned} \lambda_\alpha(i, a) &= \Lambda(a) \times Q_\alpha(i, \mathrm{BH}) \\ &= \Lambda(a) \times \left(\left(\frac{\mathrm{BH} + 1 - i}{\mathrm{BH} + 1} \right)^\alpha - \left(\frac{\mathrm{BH} - i}{\mathrm{BH} + 1} \right)^\alpha \right), \end{aligned} \tag{4.4}$$

with $i = 0, 1, \ldots, \mathrm{BH}$, $a \in \mathrm{AH}$, and for any parameter $\alpha > 0$.
The expression of $Q_\alpha(i, \mathrm{BH})$ is similar to that of the RIM quantifiers proposed in [137], after reflection and translation. To the best of our knowledge, this is the first time that a modified version of the RIM quantifiers, which are frequently used in decision making, is applied to simulation models for maximizing revenue.
We use $Q_\alpha(i, \mathrm{BH})$ because:

- they define a function with discrete domain and continuous values;
- they sum up to 1:
$$\sum_{i=0}^{BH} Q_\alpha(i, \mathrm{BH}) = 1,$$
for any $\alpha > 0$ and therefore can represent a discrete probability distribution or a normalized curve;
- they can model different reservation scenarios through α, from a constant curve ($\alpha = 1$) to increasing and decreasing curves, as reported in Figure 4.3;
- they provide a simple way of finding α from the ratio of walk-in users with respect to the total number of reservations, that is, $Q_\alpha(0, \mathrm{BH})$.

We assume that the reservation requests follow a non-homogeneous Poisson process with an expected value given by our parametric model:

$$\mathcal{R}^a \sim \mathrm{Poisson}(\Lambda(a)). \tag{4.5}$$

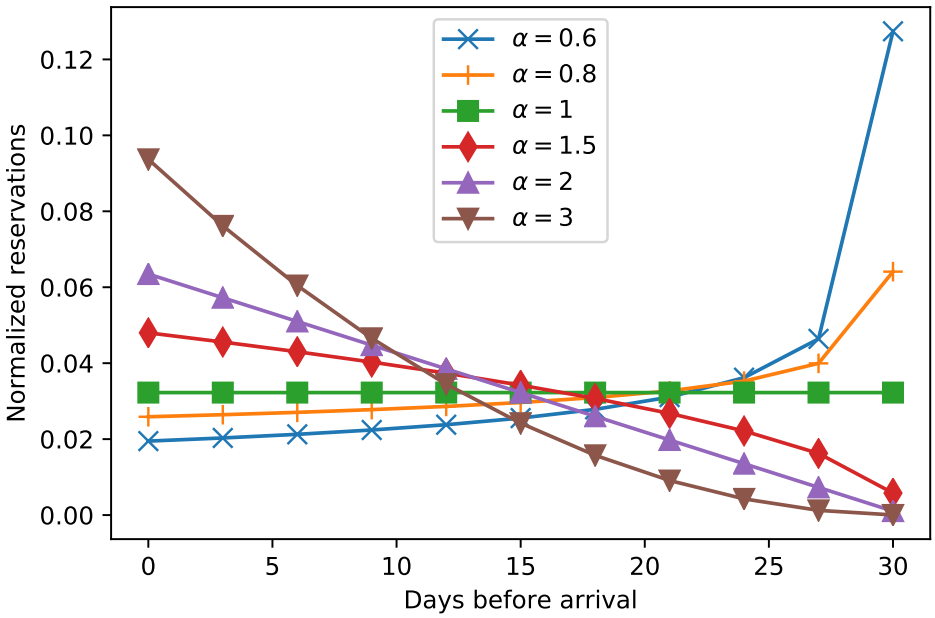


Figure 4.3: $Q_\alpha(i, \mathrm{BH})$ for $\mathrm{BH} = 30$ and for different values of α.

Therefore, reservation requests are generated for each simulated day according to the following model:

$$\begin{cases} \mathcal{R}_r^a \sim \mathrm{Poisson}(\lambda_\alpha(i, a)) & \text{if } i \leq \mathrm{BH}, \\ \mathcal{R}_r^a = 0 & \text{otherwise.} \end{cases} \tag{4.6}$$

Poisson processes are usually chosen to model arrival processes [61] and, in our context, they can represent the arrival of reservation requests with a minimum set of parameters. In [138], a binomial distribution is used, with additional constraints on the variance of samples in order to set the success probability and the number of trials. However, a binomial distribution converges to a Poisson distribution when the number of trials (e.g. customers generating requests) grows. Removing the limit on the pool of customers that can generate new reservations makes the model more realistic, since the number of possible customers is usually unbounded and independent from the capacity of the hotel.

For the estimation of $\Lambda(a)$, we assume that it is possible to estimate the expected number of reservation requests for a specific arrival day that are accepted by the customers and not canceled ($\mathcal{R}^a_{\text{accept}}$). Similarly, we assume that one has access to the expected number of reservation requests for a specific arrival day that are accepted by the customers and canceled ($\mathcal{R}^a_{\text{cancel}}$). $\mathcal{R}^a_{\text{accept}}$ can be approximated by the expected number of arrivals, while $\mathcal{R}^a_{\text{cancel}}$ can be seen as the expected number of cancellations. Both of these quantities can be estimated easily by exploiting the experience of the hotel manager. Our simulator includes also a model of the acceptance probability $\text{Pr}_{\text{accept}}(RO)$. A model of probabilities (possibly one for each admissible input) can be estimated from data retrieved by an online booking platform, where one can keep track of users that search for a room and decide to finalize the reservation or leave the website. One can also estimate the expected acceptance probability $\mathbb{E}[\text{Pr}_{\text{accept}}(RO)]$ as the expected fraction of reservation requests that are finalized by the users after the search.

Therefore, the expected total number of reservation requests (accepted or rejected) associated with one arrival day can be estimated as follows:

$$\mathbb{E}[\mathcal{R}^a] = \Lambda(a) \approx \frac{\mathcal{R}^a_{\text{accept}} + \mathcal{R}^a_{\text{cancel}}}{\mathbb{E}[\text{Pr}_{\text{accept}}(RO)]}. \tag{4.7}$$

4.3.2 Simulation of nights and rooms

Let $nights^a$ be the expected number of nights for a reservation associated with an arrival day a. Analogously, $rooms^a$ is the expected number of rooms. $max\text{-}nights^a$ and $max\text{-}rooms^a$ represent the limits imposed by the hotel manager. Since each reservation request includes at least one night and one room, we model the discrete probability distribution of the number of additional

nights/rooms as follows:

$$\Pr(X-1=k)=\int_{\frac{k}{\max(X)}}^{\frac{k+1}{\max(X)}}\frac{(1-x)^{\frac{\max(X)}{\text{avg}(X)-0.5}-2}}{\text{B}(1,\frac{\max(X)}{\text{avg}(X)-0.5}-1)}dx, \tag{4.8}$$

where X is the number of nights/rooms, $X-1$ is the number of additional nights/rooms, $\max(X)$ is either *max-nights*^{a} or *max-rooms*^{a}, and $\text{avg}(X)$ is either *nights*^{a} or *rooms*^{a}. $k=0,1,\ldots,\max(X)-1$, and $\text{B}(\alpha,\beta)$ is the Beta function with parameters α and β.

The previously defined distribution is a discrete analogue of a (continuous) Beta distribution with $\alpha=1$ and $\beta=\frac{\max(X)}{\text{avg}(X)-0.5}-1$. The value of α is chosen so as to have a distribution with an exponential-decay profile, which is similar to the distribution seen in [138]. β is chosen so as to have an expected value approximately equal to $\text{avg}(X)-1$. This is achieved by imposing the equality of the expected value of the (continuous) Beta distribution, which is $\frac{\alpha}{\alpha+\beta}$, to the expected number of additional nights/rooms rescaled to $[0,1]$, which is $\frac{\text{avg}(X)-0.5}{\max(X)}$. We consider a correction of 0.5 (found by numerical experimentation) to account for the discretization error and to position rescaled expected values in the middle of the discretization interval. Experiments show that the maximum error between the expected values and the empirical averages of the discrete analogue with $\max(X)=5$ is at most 0.33, for expected values equal to $0,0.1,0.2,\ldots,\max(X)-1$.

Even though modeling the length of stay or the number of rooms as Bernoulli or Poisson processes provides a simpler and exact way of imposing the expected value, it is not applicable to our context, which cannot be reduced to a coin toss or to an arrival process. In the literature, the Beta distribution is often used to model unknown probability distributions, with shapes that can be controlled by the parameters α and β. By building a discrete analogue of a Beta distribution, we can exploit its macroscopic features to obtain a realistic model of the variable of interest. A similar model can be defined

also for *group* reservations, which usually follow a different distribution from that of the length of stay of *normal* reservations. This can be easily achieved by considering a different value for $\text{avg}(X)$.
By following (4.8), an instance of the random variable X, which is either RR_{los} or RR_{size}, is generated as follows:

$$X = 1 + \lfloor Y \times \max(X) \rfloor, \tag{4.9}$$

where $Y \sim \text{Beta}(1, \frac{\max(X)}{\text{avg}(X)-0.5} - 1)$.

4.3.3 Simulation of cancellations

Under the same assumptions of Section 4.3.1, and by analogy to (4.1), the probability that a reservation is canceled during its lifetime can be seen as the summation of the probabilities that a reservation is canceled exactly on a specific day within its lifetime:

$$\Pr_{\text{cancel}}(R) = \Omega(a) = \sum_{i=0}^{R_{\text{TTA}}} \omega(i, a), \tag{4.10}$$

where $\omega(i, a)$ is the probability that R is canceled exactly i days before the arrival day a, with i within its lifetime.

By analogy to (4.4), we define each $\omega(i, a)$ by the following parametric model:

$$\begin{aligned} \omega_\alpha(i, a) &= \Omega(a) \times Q_\alpha(i, R_{\text{TTA}}) \\ &= \Omega(a) \times \left(\left(\frac{R_{\text{TTA}} + 1 - i}{R_{\text{TTA}} + 1} \right)^\alpha - \left(\frac{R_{\text{TTA}} - i}{R_{\text{TTA}} + 1} \right)^\alpha \right), \end{aligned} \tag{4.11}$$

with $i = 0, 1, \ldots, R_{\text{TTA}}$, $a = R_{\text{arr}}$, and for any parameter $\alpha > 0$.

In this context one can also find α from the fraction of cancellations that

occur on the last day ($Q_\alpha(0, R_{\text{TTA}})$), which includes the so-called *no-shows*.

As concerns $\Omega(a)$, it can be estimated as follows:

$$\Omega(a) \approx \frac{\mathcal{R}^a_{\text{cancel}}}{\mathcal{R}^a_{\text{cancel}} + \mathcal{R}^a_{\text{accept}}}, \tag{4.12}$$

with an arrival day $a = R_{\text{arr}}$.
Different stochastic cancellation scenarios can be simulated by changing $\omega_\alpha(i, a)$ through $\Omega(a)$ and α.

4.4 Dynamic Pricing and Acceptance Probability

As concerns the pricing policy, we use the model proposed in [20], which is based on a set of multipliers that lead to an increase or decrease in the average price according to the features of a reservation request. This model has already been validated on historical data and has led to promising results.

In addition, we assume that RO_{price} corresponds to the unit price for 1 room and 1 night. The unit price proposed to the customer is then computed as follows:

$$RO_{\text{price}} = price^a \cdot \xi(RR_{\text{TTA}}, RR_{\text{los}}, RR_{\text{size}}, S, \Delta, \eta), \tag{4.13}$$

where $price^a$ is the expected unit price for customers arriving on day a, and $\xi(\cdot)$ is a function of the reservation request features and of the hotel registry, with average value equal to 1. This function smoothly adjusts the price within the interval $[(1-\Delta)price^a, (1+\Delta)price^a]$, with a slope proportional to η:

$$\begin{aligned} &\xi(RR_{\text{TTA}}, RR_{\text{los}}, RR_{\text{size}}, S, \Delta, \eta) = \xi(t, l, s, S, \Delta, \eta) = \\ &= (1-\Delta) + 2\Delta \cdot \Phi(\eta \cdot (M_T(t)M_L(l)M_S(s)M_C(S) - 1)). \end{aligned} \tag{4.14}$$

$\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution, and $M_T(\cdot)$, $M_L(\cdot)$, $M_S(\cdot)$ and $M_C(\cdot)$ are functions (or *multipliers*) of the time-to-arrival, the length of stay, the number of rooms and the remaining hotel capacity at the moment the reservation request is generated, respectively. All the multipliers are defined so as they have an average value equal to 1.

The effect on the room demand of changing the unit price is modeled by the acceptance probability, which we define similarly to [138]. When the proposed price is equal to the average price of reservations with the same arrival day, the acceptance probability is set to 0.5, to model the absence of any preference about accepting or rejecting the reservation. With prices fixed to the average values, the expected number of accepted reservations is equal to half of the total number of reservation requests. The expected percentage of accepted reservations increases when the price decreases and decreases otherwise.
This phenomenon, also called *price elasticity*, is modeled by the following function:

$$\Pr_{\text{accept}}(RO) = 1 - \Phi(\rho \cdot (RO_{\text{price}} - price^a)), \tag{4.15}$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution, and ρ is a parameter that controls the slope of the function and allows one to consider different price elasticity scenarios.

4.5 Experiments and Discussion

In the following experiments, we validate our parametric models on aggregated data for a set of hotels with different capacities and average prices.
Room demand and customer flows are known to be influenced by the entire set of attractions, events, policies, food operators, travel agencies and hospitality businesses that characterize a tourism destination. To study the

effects of events or competitors on demand and price, network science can be used to analyze static and dynamic properties of all the actors involved [12]. However, since our main focus is on learning from aggregated data and not on tourism management, for simplicity we consider each hotel as a singleton independent from the context of its geographical region. We also assume that there is only one category of rooms. Even though this assumption is strict, the inclusion of the details of each room would require the definition of more complex choice models that are out of the scope of this Chapter.
The reservations are not tied to specific rooms. As an example, if a customer books 1 room for 2 days, the reservation is proposed to the customer for acceptance if for each single day there is at least 1 room available. This includes also the case of different physical rooms for different days for the same customer/reservation. Moreover, our simulations include walk-in users as well as those customers booking a room on the same day of arrival (e.g. in the morning for the night).

We do not have access to revenue records from the hotels and a direct comparison of our simulated revenue to the ground truth is not possible. However, under the hypotheses of the previous Sections, it can be shown that the total revenue is a deterministic function of the features of the reservations registered on the book and not canceled. Therefore, the goodness of our models can be evaluated by looking at their ability to generate events leading to aggregated indicators similar to those used as input, when no optimization is performed. Since our models lead to statistically consistent results, they can be effectively used as a baseline or bootstrap sample from which the search for an optimized pricing policy can start, as shown in the following Sections.

4.5.1 Setup of the experiments

We consider a monotonically decreasing reservation curve with 40% of the customers treated as walk-in users, similarly to the reservation models estimated from historical data in [138]. The goodness of this model is also confirmed by data collected by the Italian Institute of Statistics (Istat) on the features of trips[1], which show that approximately 40% of the interviewed people travel without booking. As a consequence, it is reasonable to assume that the remaining 60% of the reservations is monotonically distributed in the BH in a decreasing fashion as moving away from the walk-in day. We also assume that the maximum number of cancellations occurs on the last day, and we fix this number to 40% of the total number of cancellations. This number can be different in presence of cancellation fees, which can reduce late cancellations. However, as shown in [35], the booking behavior of customers, in presence of lenient cancellation policies, does not significantly differ from the strategies adopted when there is no cancellation policy. Moreover, the inclusion of non-lenient policies with high fees can lead to a reduction of cancellations and no-shows, with a possible increase in revenue. Since we are interested in the applicability of our approach to worst-case scenarios for the hotel manager, we do not consider cancellation fees.

BH is fixed to 180 days, the maximum number of nights for one reservation to 10, and the maximum number of rooms to 4.

As concerns the parameters of the multipliers, we set $T_0 = 30$ and $C_0 = L_0 = G_0 = 1.6$. All the remaining parameters are instead optimized by the system in order to maximize the total revenue. For a more detailed description of the parameters see [20].

η is fixed to 3, while Δ is fixed to 0.6, so as to propose prices with a maximum increase/decrease of 60% with respect to $price^a$.

[1] http://dati.istat.it/?lang=en; section: Communication, culture, trips/Trips/Trips and their characteristics; last access: March, 2019

Table 4.1: Hotels used for the tests.

Hotel ID	Stars	Rooms	Average price (€)
01	3	52	120.00
02	2	34	69.50
03	4	136	290.00
04	4	46	153.33
05	3	113	136.67
06	3	37	74.00
07	1	9	39.00
08	4	22	216.50
09	2	14	66.50
10	3	19	82.67

In the experiments, ρ is chosen so that $\Pr_{\text{accept}}(RO) \approx 1$ when there is a discount of at least 50% and $\Pr_{\text{accept}}(RO) \approx 0$ when the price increases of at least 50%. In this way we are able to simulate a realistic scenario, in which the ranges of acceptable prices for customers and hotel managers differ. In our case, customers can accept maximum variations of $\pm 50\%$, while hotel managers allow maximum variations of proposed prices of $\pm 60\%$ (through Δ in (4.13)).

We empirically show the applicability of our methodology to 10 hotels in Trento, Italy. In order to simulate multiple scenarios, we selected (and anonymized) representative hotels from the official open data of the Province of Trento[2], with different capacities, stars and average prices, as reported in Table 4.1. The information related to the average arrivals and the average number of nights per reservation is taken from the Statistics Institute of the Province of Trento (Ispat)[3]. Only monthly data up to 2016 are available.

[2]http://dati.trentino.it/dataset/esercizi-alberghieri (last access: March, 2019).
[3]http://www.statistica.provincia.tn.it, section “Annuari del Turismo" (last access: March, 2019).

No information is available about the average number of rooms per reservation, so we assumed it to be equal to 1 to make it irrelevant for the optimization. We disaggregated data on arrivals and mapped them onto each hotel according to their capacity, under the assumption that bigger hotels usually register more arrivals than smaller hotels. Since we do not have access to the complete history of each hotel, we assume that the data of 2016 are valid also in the following years.

In the experiments RH starts on July 1st, 2017, and ends on December 31st, 2018. AH starts on July 1st, 2017, and ends on January 31st, 2019. OH starts on January 1st, 2018, and ends on December 31st, 2018. To avoid bias given by strict truncation, we maximize the total revenue in one year, after a transient period of 6 months and for customers arriving up to one month later than the end of the year of interest.

For the optimization, we use an efficient implementation of CMA-ES[4], with a step size of 0.5 (for an optimal exploration-exploitation balance) and a budget of 300 iterations (for a maximum estimated optimization time of 5/6 hours). Each iteration retrieves the total revenue as the average on 20 simulation runs, all with the same parameter configuration, for a total of 6000 simulations within one optimization run. For each hotel, the optimization process is repeated 10 times.

To improve the efficiency and effectiveness of optimization, if one considers variable costs and profit, a possible approach is given by multi-objective optimization to take into account the revenue generated by arrivals as well as the costs associated with occupancy. Recent examples of multi-objective optimization heuristics are presented in [26]. Another approach is given by nested optimization schemes [60]. These techniques can lead to improved efficiency by reducing our multidimensional problem, which depends on different dimensions like the time to arrival and the length of stay, to a family

[4]Code available at http://beniz.github.io/libcmaes (last access: March, 2019).

of one-dimensional subproblems. However, the comparison of different optimization methods is out of the scope of this Chapter, which focuses mainly on the challenge of learning parametric models through simulations on aggregated data. We selected CMA-ES because it is one of the state-of-the-art algorithms for black-box optimization and in particular it is the technique used by [20], from which we developed our pricing model.

4.5.2 Results on arrivals, occupancy and revenue

For each hotel, we used a one-sample t-test for testing the equivalence between the average total number of arrivals used as input and the same indicator produced by simulations with non-optimized prices. With a significance level $\alpha = 0.01$ and a total of 200 runs for each hotel, the equivalence of input and output is empirically proved, with a computed statistical power of 1 for all the hotels. Therefore, we can assume that the implementation of our models is correct and unbiased. This allows one to consider different relationships between the same set of inputs and outputs, with stochastic functions that can better represent realistic scenarios characterized by uncertainty.

Independently from the specific hotel and scenario, optimized pricing policies based on our parametric models always lead to statistically significant improvements. In Table 4.2 we report the results on customer arrivals, occupancy and revenue as the percentage increase led by the optimized pricing model with respect to the configuration with the multipliers equal to 1.
In our context, a unit of occupancy corresponds to the so-called *room-night*, which is a room occupied for one night. Results are expressed in terms of averages and standard errors, and they are statistically significant according to the two-tailed unequal variances t-test [132], with a significance level $\alpha = 0.01$.

Results are promising for all the hotels, with a minimum of 12.8% increase in revenue, 37.7% in occupancy and 38.2% in arrivals. The maximum

Table 4.2: Percentage increase in arrivals, occupancy (as room-nights) and revenue after optimization. Maximum and minimum values are in bold.

Hotel ID	Arrivals	Occupancy	Revenue
01	48.2±0.5	47.6±0.6	18.4±0.6
02	50.6±0.6	50.6±0.7	20.4±0.7
03	51.6±0.3	52.6±0.3	21.6±0.3
04	44.0±0.5	44.2±0.6	17.8±0.6
05	**55.5±0.3**	**55.2±0.4**	**23.1±0.4**
06	46.9±0.6	45.8±0.6	17.8±0.6
07	**38.2±1.1**	**37.7±1.3**	**12.8±1.2**
08	43.2±0.7	42.0±0.8	18.0±0.8
09	42.0±0.9	41.8±1.0	17.7±1.0
10	41.8±0.8	40.5±0.9	17.3±0.9

increase in revenue is reached for Hotel 05, with a value of 23.1%. The minimum values are reached for small hotels, where the limited number of rooms leads to fewer arrivals and then relatively low revenues. In this context, there is also more variability, since the hotel can become full with few reservations, thus leading to the rejection of more requests.

Experiments suggest that higher revenues can be obtained for medium and big hotels, where the system exploits the capacity of the hotel to increase the number of arrivals. The time series of the average daily revenue during the year of interest for the best and worst scenarios are reported in Figure 4.4.

For Hotel 05, it is evident that the time series produced by the optimized model is significantly higher than that produced without optimization. In this case, there is less chance of having a loss in revenue because of an op-

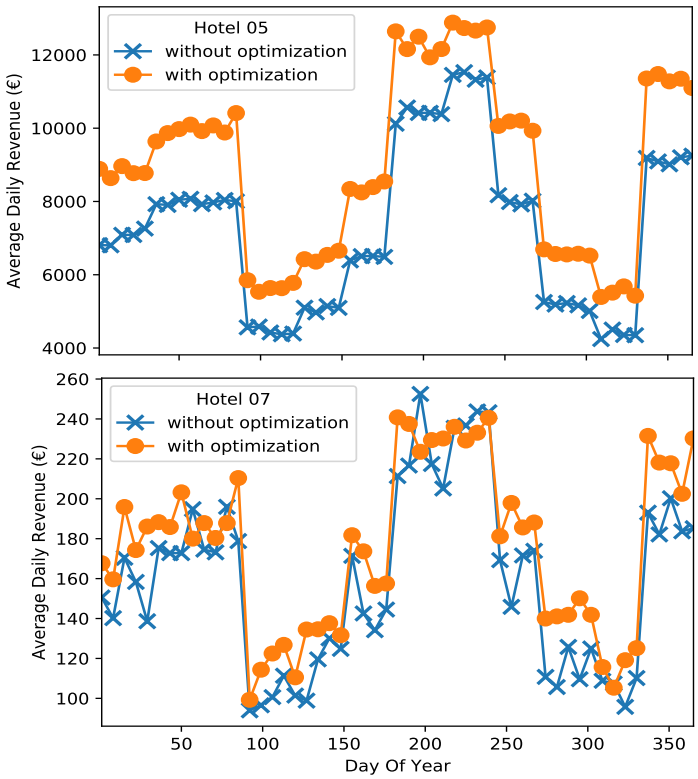


Figure 4.4: Average daily revenue for Hotel 05 and Hotel 07 (one value per week).

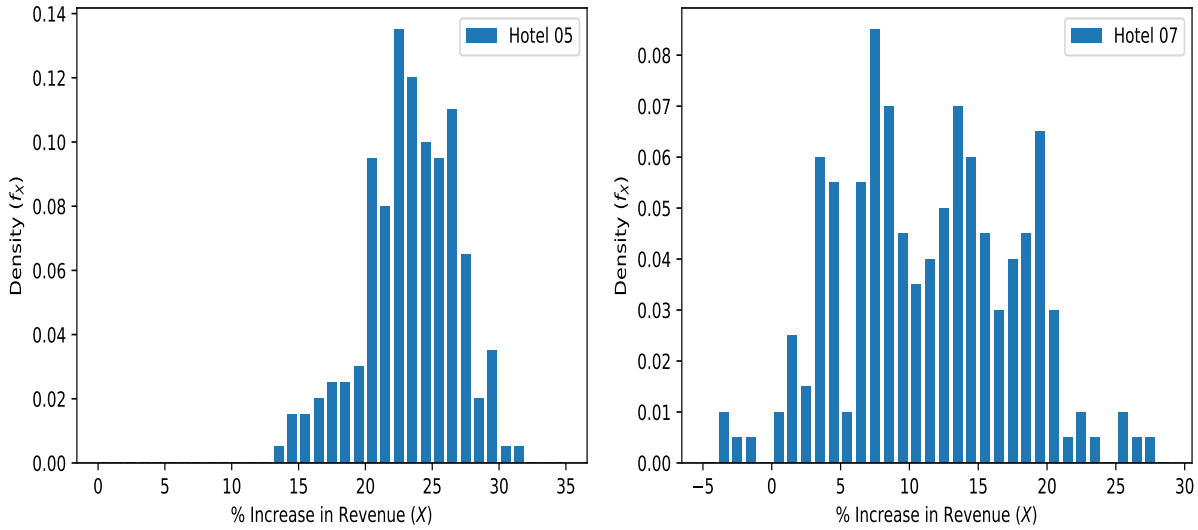


Figure 4.5: Estimated distributions of increase in revenue after optimization for Hotel 05 and Hotel 07.

timistic configuration found during the optimization process. For Hotel 07, the two time series are not significantly different because of the higher uncertainty caused by the small dimension of the hotel. This leads to higher risk and to the possibility of having a loss, even though this can happen with a relatively low probability (≈ 0.03), as it is evident from the distribution of the increase in revenue in Figure 4.5. These results are in accordance with the expected behavior of non-homogeneous Poisson distributions, whose coefficient of variation decreases as the expected value increases. In the context of hotel demand, this property implies that for small hotels, which can accommodate a limited number of guests and therefore are characterized by less arrivals, the coefficient of variation is higher than that of large hotels. As a consequence, the increased variability for small hotels leads to higher risk of losses, as empirically shown by our results.

Independently from the numerical values, the case study and the experiments reported in this Chapter support our claim that learning, in the extreme case when one has no access to individual samples, is still possible. However, there is the need of having access to the knowledge of domain experts or previous research results, for example in the form of aggregated data. In addition, it is crucial to develop robust techniques that take into account the high level of uncertainty of the modeled phenomena in order to define risk-averse methodologies based on Monte Carlo simulations and statistical significance and power.

Chapter 5

Conclusion

In this dissertation we focused on the effects of noise and uncertainty on machine learning. We briefly reviewed some state-of-the-art solutions to build robust algorithms as well as to detect and reduce noise before the actual learning phase. We then addressed the problems arising in three contexts characterized by different volumes of samples.

When the number of samples is large enough to learn an accurate model, high dimensionality, feature redundancy and feature irrelevance can still deceive the training and lead to poor generalization performance. In this context, we proposed a novel approach for feature selection based on the minimization of the neighborhood entropy, which is inversely proportional to the mutual information between the features and the output variable. We developed an algorithm robust to noise and to class imbalance by using a greedy procedure based on approximated nearest neighbors and locality-sensitive hashing. Experimental results showed that our technique usually leads to better classification accuracy, at the expense of more computation time. In addition, our method tends to select features with a better order, leading to a better classification accuracy for fewer features, even in the presence of high levels of noise and class imbalance. Possible future research directions include a comparison between our filter method and embedded methods like

autoencoders, as well as the study and implementation of techniques able to reduce the CPU time.

As concerns problems characterized by data scarcity and high dimensionality, we showed that feature selection is not always beneficial for learning. In the context of salary prediction in the IT job market, we empirically demonstrated that ad-hoc feature engineering and ensemble learning are effective solutions to handle the uncertainty caused by the limited number of samples. We compared several models including logistic regression, nearest-neighbors classifiers, neural networks, support vector machines, random forests, boosting machines and majority-vote ensembles based on all or part of them. Experiments showed that our heterogeneous ensembles lead to superior results in terms of accuracy, precision and recall. As future work, it would be interesting to study stacked heterogeneous ensembles including embedded feature selection methods.

Lastly, we studied the extreme case when individual samples are not available and only aggregated data are accessible. In this situation, traditional instance-based learning is not possible. Our solution, in the context of dynamic pricing for hotel revenue management, includes the definition of parametric and stochastic models, and the formulation of learning as a simulation-based black-box optimization problem. Useful insights can be retrieved by analyzing different scenarios through a limited set of meaningful parameters. Experiments on real aggregated data showed that, even with no access to individual historical records, our proposed methodology effectively handles the inherent stochasticity of room demand and allows hotel managers to conduct rigorous risk analysis. Future work will focus on the extension of our parametric models to include multiple categories of customers and rooms as well as the effects of competitors on demand and price.

Bibliography

[1] Y. Abbound, A. Boyer, and A. Brun. Predict the emergence: Application to competencies in job offers. In *Tools with Artificial Intelligence (ICTAI), 2015 IEEE 27th International Conference on*, pages 612–619, Nov 2015.

[2] J. Abellán and A. R. Masegosa. Bagging decision trees on data sets with classification noise. In *International Symposium on Foundations of Information and Knowledge Systems*, pages 248–265. Springer, 2010.

[3] A. O. Akyuz, M. Uysal, B. A. Bulbul, and M. O. Uysal. Ensemble approach for time series analysis in demand forecasting: Ensemble learning. In *INnovations in Intelligent SysTems and Applications (INISTA), 2017 IEEE International Conference on*, pages 7–12. IEEE, 2017.

[4] S. T. Al-Otaibi and M. Ykhlef. A survey of job recommender systems. *International Journal of the Physical Sciences*, 7(29):5127–5142, 2012.

[5] A. Alarcón-Paredes, G. A. Alonso, E. Cabrera, and R. Cuevas-Valencia. Simultaneous gene selection and weighting in nearest neighbor classifier for gene expression data. In *International Conference on Bioinformatics and Biomedical Engineering*, pages 372–381. Springer, 2017.

[6] J. Alcalá, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *J. Multiple-Valued Logic Soft Comput.*, 17(2-3):255–287, 2010. http://sci2s.ugr.es/keel/datasets.php [accessed Jan. 2018].

[7] N. Almalis, G. Tsihrintzis, and N. Karagiannis. *Research and Development in Intelligent Systems XXXII: Incorporating Applications and Innovations in Intelligent Systems XXIII*, chapter A New Content-Based Recommendation Algorithm for Job Recruiting, pages 393–398. Springer International Publishing, Cham, 2015.

[8] F. Amato, R. Boselli, M. Cesarini, F. Mercorio, M. Mezzanzanica, V. Moscato, F. Persia, and A. Picariello. Challenge: Processing web texts for classifying job offers. In *Semantic Computing (ICSC), 2015 IEEE International Conference on*, pages 460–463, Feb 2015.

[9] R. R. Andrawis and A. F. Atiya. A new Bayesian formulation for Holt's exponential smoothing. *Journal of Forecasting*, 28(3):218–234, 2009.

[10] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923, 1998.

[11] H. A. Aziz, M. Saleh, M. H. Rasmy, and H. Elshishiny. Dynamic room pricing model for hotel revenue management systems. *Egyptian Informatics Journal*, 12(3):177–183, 2011.

[12] R. Baggio. Network science and tourism–the state of the art. *Tourism Review*, 72(1):120–131, 2017.

[13] T. K. Baker and D. A. Collier. A comparative revenue analysis of hotel yield management heuristics. *Decision Sciences*, 30(1):239–263, 1999.

[14] R. Barandela and E. Gasca. Decontamination of training samples for supervised pattern recognition methods. In *Joint IAPR International Workshops on Statistical Techniques*

in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), pages 621–630. Springer, 2000.

[15] L. Barber. *E-recruitment Developments*. Institute for Employment Studies, 2006.

[16] G. E. Batista, R. C. Prati, and M. C. Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, 6(1):20–29, 2004.

[17] R. Battiti. Accelerated backpropagation learning: Two optimization methods. *Complex systems*, 3(4):331–342, 1989.

[18] R. Battiti. Using the mutual information for selecting features in supervised neural net learning. *IEEE Trans. Neural Netw.*, 5(4):537–550, 1994.

[19] R. Battiti and M. Brunato. *The LION way. Machine Learning* plus *Intelligent Optimization*. LIONlab, University of Trento, Italy, December 2017.

[20] A. E.-M. Bayoumi, M. Saleh, A. F. Atiya, and H. A. Aziz. Dynamic pricing for hotel revenue management using price multipliers. *Journal of Revenue and Pricing Management*, 12(3):271–285, 2013.

[21] J. Beemer, K. Spoon, L. He, J. Fan, and R. A. Levine. Ensemble learning for estimating individualized treatment effects in student success studies. *International Journal of Artificial Intelligence in Education*, pages 1–21, 2017.

[22] D. Bertsimas and S. De Boer. Simulation-based booking limits for airline revenue management. *Operations Research*, 53(1):90–106, 2005.

[23] D. Bertsimas and I. Popescu. Revenue management in a dynamic network environment. *Transportation science*, 37(3):257–277, 2003.

[24] G. Bitran and R. Caldentey. An overview of pricing models for revenue management. *Manufacturing & Service Operations Management*, 5(3):203–229, 2003.

[25] G. R. Bitran and S. V. Mondschein. An application of yield management to the hotel industry considering multiple day stays. *Operations research*, 43(3):427–443, 1995.

[26] A. Blot, M.-É. Kessaci, and L. Jourdan. Survey and unification of local search techniques in metaheuristics for multi-objective combinatorial optimisation. *Journal of Heuristics*, 24(6):853–877, Dec 2018.

[27] D. Bouzas, N. Arvanitopoulos, and A. Tefas. Graph embedded nonparametric mutual information for supervised dimensionality reduction. *IEEE Trans. Neural Netw. Learn. Syst.*, 26(5):951–963, 2015.

[28] A. Bozzon, M. Brambilla, S. Ceri, M. Silvestri, and G. Vesci. Choosing the right crowd: expert finding in social networks. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 637–648. ACM, 2013.

[29] C. E. Brodley and M. A. Friedl. Identifying mislabeled training data. *Journal of artificial intelligence research*, 11:131–167, 1999.

[30] M. Brunato and R. Battiti. X-mifs: Exact mutual information for feature selection. In *Proc. Intl. Joint Conf. on Neural Netw.*, pages 3469–3476, 2016.

[31] D. Buhalis and R. Law. Progress in information technology and tourism management: 20 years on and 10 years after the internet – the state of etourism research. *Tourism management*, 29(4):609–623, 2008.

[32] F. Bulut and M. F. Amasyali. Locally adaptive k parameter selection for nearest neighbor classifier: one nearest cluster. *Pattern Analysis and Applications*, 20(2):415–425, 2017.

[33] R. Campos, M. Arrazola, and J. de Hevia. Online job search in the spanish labor market. *Telecommunications Policy*, 38(11):1095 – 1116, 2014.

[34] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[35] C.-C. Chen, Z. Schwartz, and P. Vargas. The search for the best deal: How hotel cancellation policies affect the search and booking decisions of deal-seeking customers. *International Journal of Hospitality Management*, 30(1):129–135, 2011.

[36] C.-F. Chien and L.-F. Chen. Data mining to improve personnel selection and enhance human capital: A case study in high-technology industry. *Expert Systems with Applications*, 34(1):280 – 290, 2008.

[37] T. Y. Choi and V. Cho. Towards a knowledge discovery framework for yield management in the Hong Kong hotel industry. *International Journal of Hospitality Management*, 19(1):17–31, 2000.

[38] A. K. Chowdhury, D. Tjondronegoro, V. Chandran, and S. G. Trost. Ensemble methods for classification of physical activities from wrist accelerometry. *Medicine and science in sports and exercise*, 49(9):1965, 2017.

[39] K. L. Clarkson. Fast algorithms for the all nearest neighbors problem. In *IEEE Symp. on Foundations of Computer Science*, pages 226–232, 1983.

[40] S. Corchs, E. Fersini, and F. Gasparini. Ensemble learning on visual and textual data for social image emotion classification. *International Journal of Machine Learning and Cybernetics*, pages 1–14, 2017.

[41] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. 20th annual symposium on Computational geometry*, pages 253–262. ACM, 2004.

[42] B. Denizci Guillet and I. Mohammed. Revenue management research in hospitality and tourism: A critical review of current literature and suggestions for future research. *International Journal of Contemporary Hospitality Management*, 27(4):526–560, 2015.

[43] S. Diamond, V. Sitzmann, S. Boyd, G. Wetzstein, and F. Heide. Dirty pixels: Optimizing image classification architectures for raw sensor data. *arXiv preprint arXiv:1701.06487*, 2017.

[44] T. G. Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[45] T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, 40(2):139–157, 2000.

[46] S. Dodge and L. Karam. Understanding how image quality affects deep neural networks. In *Quality of Multimedia Experience (QoMEX), 2016 Eighth International Conference on*, pages 1–6. IEEE, 2016.

[47] S. Dodge and L. Karam. Quality resilient deep neural networks. *arXiv preprint arXiv:1703.08119*, 2017.

[48] M. Ekström, P.-A. Esseen, B. Westerlund, A. Grafström, B. Jonsson, and G. Ståhl. Logistic regression for clustered data from environmental monitoring programs. *Ecological Informatics*, 2017.

[49] P. A. Estévez, M. Tesmer, C. A. Perez, and J. M. Zurada. Normalized mutual information feature selection. *IEEE Trans. Neural Netw.*, 20(2):189–201, 2009.

[50] E. Faliagka, K. Ramantas, A. K. Tsakalidis, M. Viennas, E. Kafeza, and G. Tzimas. An integrated e-recruitment system for cv ranking based on ahp. In *WEBIST*, pages 147–150, 2011.

[51] E. Faliagka, A. Tsakalidis, and G. Tzimas. An integrated e-recruitment system for automated personality mining and applicant ranking. *Internet Research*, 22(5):551–568, 2012.

[52] R. Fano and D. Hawkins. Transmission of information. *Am. J. Phys.*, 29(11):793–794,

1961.

[53] G. Figueira and B. Almada-Lobo. Hybrid simulation–optimization methods: A taxonomy and discussion. *Simulation Modelling Practice and Theory*, 46:118–134, 2014.

[54] D. H. Fisher and K. B. McKusick. An empirical comparison of id3 and back-propagation. In *IJCAI*, pages 788–793, 1989.

[55] B. Frénay, G. Doquire, and M. Verleysen. Estimating mutual information for feature selection in the presence of label noise. *Computational Statistics & Data Analysis*, 71:832–848, 2014.

[56] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *J. Am. Stat. Assoc.*, 32(200):675–701, 1937.

[57] E. S. Gardner Jr. Exponential smoothing: The state of the art—part ii. *International journal of forecasting*, 22(4):637–666, 2006.

[58] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. *Proc. 25th VLDB Conf.*, 1999.

[59] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet. A survey on ensemble learning for data stream classification. *ACM Computing Surveys (CSUR)*, 50(2):23, 2017.

[60] V. Grishagin, R. Israfilov, and Y. Sergeyev. Convergence conditions and numerical comparison of global optimization methods based on dimensionality reduction schemes. *Applied Mathematics and Computation*, 318:270–280, 2018.

[61] P. Grube, F. Núñez, and A. Cipriano. An event-driven simulator for multi-line metro systems and its application to Santiago de Chile metropolitan rail network. *Simulation Modelling Practice and Theory*, 19(1):393–405, 2011.

[62] J. Guadix, P. Cortés, L. Onieva, and J. Muñuzuri. Technology revenue management system for customer groups in hotels. *Journal of Business Research*, 63(5):519–527, 2010.

[63] M. A. Hall and L. A. Smith. Feature subset selection: a correlation based filter approach. In *Proc. Intl. Conf. Neural Inform. Processing Intell. Inform. Syst.*, pages 855–858. Springer, 1997.

[64] N. Hansen. The CMA evolution strategy: a comparing review. In *Towards a new evolutionary computation*, pages 75–102. Springer, 2006.

[65] N. Hansen. Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, pages 2389–2396. ACM, 2009.

[66] S. Hariharan, S. Tirodkar, A. Porwal, A. Bhattacharya, and A. Joly. Random forest-based prospectivity modelling of greenfield terrains using sparse deposit data: An example from the tanami region, western australia. *Natural Resources Research*, pages 1–19, 2017.

[67] B. Heap, A. Krzywicki, W. Wobcke, M. Bain, and P. Compton. *PRICAI 2014: Trends in Artificial Intelligence: 13th Pacific Rim International Conference on Artificial Intelligence, Gold Coast, QLD, Australia, December 1-5, 2014. Proceedings*, chapter Combining Career Progression and Profile Matching in a Job Recommender System, pages 396–408. Springer International Publishing, Cham, 2014.

[68] A. B. Holm. E-recruitment: Towards an ubiquitous recruitment process and candidate relationship management. *German Journal of Human Resource Management*, 26(3):241–259, 2012.

[69] W. Hong, S. Zheng, H. Wang, and J. Shi. A job recommender system based on user clustering. *Journal of Computers*, 8(8):1960–1967, 2013.

[70] J. J. Horton. The effects of algorithmic labor market recommendations: Evidence from a field experiment. *Available at SSRN 2346486*, 2015.

[71] H. Hosseini, B. Xiao, and R. Poovendran. Google's cloud vision api is not robust to noise. In

Machine Learning and Applications (ICMLA), 2017 16th IEEE International Conference on, pages 101–105. IEEE, 2017.

[72] C. Hou, F. Nie, X. Li, D. Yi, and Y. Wu. Joint embedding learning and sparse regression: A framework for unsupervised feature selection. *IEEE Trans. Cybern.*, 44(6):793–804, 2014.

[73] S. Ivanov. *Hotel revenue management: From theory to practice*. Zangador, 2014.

[74] P. Jafari and F. Azuaje. An assessment of recently published gene expression data analyses: reporting experimental design and statistical factors. *BMC Med. Inform. Decis. Mak.*, 6(1):27, 2006.

[75] H. Jantan, A. R. Hamdan, and Z. A. Othman. Knowledge discovery techniques for talent forecasting in human resource application. *World Academy of Science, Engineering and Technology*, 50:775–783, 2009.

[76] G. H. John, R. Kohavi, K. Pfleger, et al. Irrelevant features and the subset selection problem. In *Machine learning: Proc. of the 11th Intl. Conf.*, pages 121–129, 1994.

[77] S. Karahan, M. K. Yildirum, K. Kirtac, F. S. Rende, G. Butun, and H. K. Ekenel. How image degradations affect deep cnn-based face recognition? In *Biometrics Special Interest Group (BIOSIG), 2016 International Conference of the*, pages 1–5. IEEE, 2016.

[78] T. M. Khoshgoftaar, S. Zhong, and V. Joshi. Enhancing software quality estimation using ensemble-classifier based noise filtering. *Intelligent Data Analysis*, 9(1):3–27, 2005.

[79] K. Kira and L. A. Rendell. A practical approach to feature selection. In *Proc. 9th Intl. Workshop on Machine learning*, pages 249–256, 1992.

[80] J. Kittler. Feature selection and extraction. *Handbook of pattern recognition and image processing*, pages 59–83, 1986.

[81] A. J. Kleywegt. An optimal control problem of dynamic pricing. *School of Industrial and Systems Engineering, Georgia Institute of Technology (2001)*, 2001.

[82] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artif. Intell.*, 97(1):273–324, 1997.

[83] I. Kononenko, E. Šimec, and M. Robnik-Šikonja. Overcoming the myopia of inductive learning algorithms with relieff. *Appl. Intell.*, 7(1):39–55, 1997.

[84] A. Kraskov, H. Stögbauer, and P. Grassberger. Estimating mutual information. *Physical Review E*, 69(6), 2004.

[85] P. Kuhn and H. Mansour. Is internet job search still ineffective? *The Economic Journal*, 124(581):1213–1233, 2014.

[86] N. Kwak and C.-H. Choi. Input feature selection by mutual information based on parzen window. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(12):1667–1671, 2002.

[87] K.-K. Lai and W.-L. Ng. A stochastic approach to hotel revenue optimization. *Computers & Operations Research*, 32(5):1059–1072, 2005.

[88] N. D. Lawrence and B. Schölkopf. Estimating a kernel fisher discriminant in the presence of label noise. In *ICML*, volume 1, pages 306–313. Citeseer, 2001.

[89] L. Lefakis and F. Fleuret. Jointly informative feature selection. In *AISTATS*, pages 567–575, 2014.

[90] M. Lichman. UCI machine learning repository, 2013.

[91] R. Linsker. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988.

[92] S. Liu, K. K. Lai, and S. Wang. Booking models for hotel revenue management considering multiple-day stays. 2:78–91, 02 2008.

[93] Y. Liu, C. Jiang, and H. Zhao. Using contextual features and multi-view ensemble learning in product defect identification from online discussion forums. *Decision Support Systems*,

2017.

[94] C. Mang. Online job search and matching quality. Technical report, Ifo Working Paper, 2012.

[95] G. Mani. Some approaches to handle noise in concept learning. *International journal of man-machine studies*, 36(2):167–181, 1992.

[96] J. Mata, I. de Miguel, R. J. Duran, N. Merayo, S. K. Singh, A. Jukan, and M. Chamania. Artificial intelligence (ai) methods in optical networks: A comprehensive survey. *Optical Switching and Networking*, 2018.

[97] J. I. McGill and G. J. Van Ryzin. Revenue management: Research overview and prospects. *Transportation science*, 33(2):233–256, 1999.

[98] P. Melville, N. Shah, L. Mihalkova, and R. J. Mooney. Experiments on ensembles with missing and noisy data. In *International Workshop on Multiple Classifier Systems*, pages 293–302. Springer, 2004.

[99] M. Muja and D. G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(11):2227–2240, 2014.

[100] D. F. Nettleton, A. Orriols-Puig, and A. Fornells. A study of the effect of different types of noise on the precision of supervised learning techniques. *Artificial intelligence review*, 33(4):275–306, 2010.

[101] F. Nie, H. Huang, X. Cai, and C. H. Ding. Efficient and robust feature selection via joint l2,1-norms minimization. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Adv. Neural Inf. Process. Syst. 23*, pages 1813–1821. Curran Associates, Inc., 2010.

[102] S. Palaniappan, T. Rajinikanth, and A. Govardhan. Spatial data analysis using various tree classifiers ensembled with adaboost approach. In *Emerging Trends in Electrical, Communications and Information Technologies: Proceedings of ICECIT-2015*, pages 165–174. Springer, 2017.

[103] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.

[104] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12(Oct):2825–2830, 2011. `http://scikit-learn.org`.

[105] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1226–1238, 2005.

[106] B. T. Pham, D. T. Bui, H. R. Pourghasemi, P. Indra, and M. Dholakia. Landslide susceptibility assesssment in the uttarakhand area (india) using gis: a comparison study of prediction capability of naïve bayes, multilayer perceptron neural networks, and functional trees methods. *Theoretical and Applied Climatology*, 128(1-2):255–273, 2017.

[107] F. Provost, C. Hibert, and J.-P. Malet. Automatic classification of endogenous landslide seismicity using the random forest supervised classifier. *Geophysical Research Letters*, 44(1):113–120, 2017.

[108] J. R. Quinlan. The effect of noise on concept learning. *Machine learning: An artificial intelligence approach*, 2:149–166, 1986.

[109] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[110] R. Schlögel, I. Marchesini, M. Alvioli, P. Reichenbach, M. Rossi, and J.-P. Malet. Optimizing landslide susceptibility zonation: Effects of dem spatial resolution and slope unit delineation on logistic regression models. *Geomorphology*, 2017.

[111] N. Segata, E. Blanzieri, S. J. Delany, and P. Cunningham. Noise reduction for instance-based learning with a local maximal margin approach. *Journal of Intelligent Information Systems*, 35(2):301–331, 2010.

[112] V. Senthil Kumaran and A. Sankar. Towards an automated system for intelligent screening of candidates for recruitment using ontology mapping (expert). *International Journal of Metadata, Semantics and Ontologies*, 8(1):56–64, 2013.

[113] U. Shaham, Y. Yamada, and S. Negahban. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing*, 2018.

[114] A. A. Shanab, T. M. Khoshgoftaar, and R. Wald. Robustness of threshold-based feature rankers with data sampling on noisy and imbalanced data. In *FLAIRS Conference*, 2012.

[115] C. Silpa-Anan and R. Hartley. Optimised kd-trees for fast image descriptor matching. In *CVPR IEEE Conf.*, pages 1–8, 2008.

[116] A. Singh, C. Rose, K. Visweswariah, V. Chenthamarakshan, and N. Kambhatla. Prospect: A system for screening candidates for recruitment. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 659–668, New York, NY, USA, 2010. ACM.

[117] N. Sivaram and K. Ramar. Applicability of clustering and classification algorithms for recruitment data mining. *International Journal of Computer Applications*, 4(5):23–28, 2010.

[118] M. Sprenger, S. Schemm, R. Oechslin, and J. Jenkner. Nowcasting foehn wind events using the adaboost machine learning algorithm. *Weather and Forecasting*, 32(3):1079–1099, 2017.

[119] D. Stefanovic, N. Stefanovic, and B. Radenkovic. Supply network modelling and simulation methodology. *Simulation Modelling Practice and Theory*, 17(4):743–766, 2009.

[120] K. T. Talluri and G. J. Van Ryzin. *The theory and practice of revenue management*, volume 68. Springer Science & Business Media, 2006.

[121] H. Tao, C. Hou, F. Nie, Y. Jiao, and D. Yi. Effective discriminative feature selection with nontrivial solution. *IEEE Trans. Neural Netw. Learn. Syst.*, 27(4):796–808, 2016.

[122] L. F. Thompson, P. W. Braddy, and K. L. Wuensch. E-recruitment and the benefits of organizational web appeal. *Computers in Human Behavior*, 24(5):2384 – 2398, 2008. Including the Special Issue: Internet Empowerment.

[123] J. Tian, M. H. Azarian, M. Pecht, G. Niu, and C. Li. An ensemble learning-based fault diagnosis method for rotating machinery. In *Prognostics and System Health Management Conference (PHM-Harbin), 2017*, pages 1–6. IEEE, 2017.

[124] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Series B Stat. Methodol.*, 58(1):267–288, 1996.

[125] I. Tomek. Two modifications of cnn. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-6(11):769–772, 1976.

[126] K. Torkkola. Feature extraction by non-parametric mutual information maximization. *J. Mach. Learn. Res.*, 3(3):1415–1438, 2003.

[127] P. M. Vaidya. An o(nlogn) algorithm for the all-nearest-neighbors problem. *Disc. Comp. Geom.*, 4(2):101–115, 1989.

[128] K. R. Varshney, V. Chenthamarakshan, S. W. Fancher, J. Wang, D. Fang, and A. Mojsilović. Predicting employee expertise for talent management in the enterprise. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1729–1738, New York, NY, USA, 2014. ACM.

[129] R. Wang, F. Nie, R. Hong, X. Chang, X. Yang, and W. Yu. Fast and orthogonal locality preserving projections for dimensionality reduction. *IEEE Trans. Image Process.*, 26(10):5019–5030, 2017.

[130] R. Wang, F. Nie, X. Yang, F. Gao, and M. Yao. Robust 2DPCA with non-greedy l1-norm maximization for image analysis. *IEEE Trans. Cybern.*, 45(5):1108–1112, 2015.

[131] L. R. Weatherford and S. E. Kimes. A comparison of forecasting methods for hotel revenue management. *International journal of forecasting*, 19(3):401–415, 2003.

[132] B. L. Welch. The generalization of student's problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35, 1947.

[133] S. Xiang, F. Nie, G. Meng, C. Pan, and C. Zhang. Discriminative least squares regression for multiclass classification and feature selection. *IEEE Trans. Neural Netw.*, 23(11):1738–1754, 2012.

[134] Z. Xiang, V. P. Magnini, and D. R. Fesenmaier. Information technology and consumer behavior in travel and tourism: Insights from travel planning using the internet. *Journal of Retailing and Consumer Services*, 22:244–249, 2015.

[135] J. Xu, X. Liu, Z. Huo, C. Deng, F. Nie, and H. Huang. Multi-class support vector machine via maximizing multi-class margins. In *The 26th International Joint Conference on Artificial Intelligence (IJCAI 2017)*, 2017.

[136] J. Xu, B. Tang, H. He, and H. Man. Semisupervised feature selection based on relevance and redundancy criteria. *IEEE Trans. Neural Netw. Learn. Syst.*, 28(9):1974–1984, 2017.

[137] R. R. Yager. Quantifier guided aggregation using OWA operators. *International Journal of Intelligent Systems*, 11(1):49–73, 1996.

[138] A. Zakhary, A. F. Atiya, H. El-Shishiny, and N. E. Gayar. Forecasting hotel arrivals and occupancy using Monte Carlo simulation. *Journal of Revenue and Pricing Management*, 10(4):344–366, 2011.

[139] A. Zakhary, N. El Gayar, and A. F. Atiya. A comparative study of the pickup method and its variations using a simulated hotel reservation data. *ICGST international journal on artificial intelligence and machine learning*, 8:15–21, 2008.

[140] F. Zaman and H. Hirose. Effect of subsampling rate on subbagging and related ensembles of stable classifiers. In *International Conference on Pattern Recognition and Machine Intelligence*, pages 44–49. Springer, 2009.

[141] D. Zhang and L. Weatherford. Dynamic pricing for network revenue management: A new approach and application in the hotel industry. *INFORMS Journal on Computing*, 29(1):18–35, 2016.

[142] J. Zhang, J. Tang, and J. Li. Expert finding in a social network. In *Advances in Databases: Concepts, Systems and Applications*, pages 1066–1069. Springer, 2007.

[143] W. Zhang, R. Rekaya, and K. Bertrand. A method for predicting disease subtypes in presence of misclassification among training samples using gene expression: application to human breast cancer. *Bioinformatics*, 22(3):317–325, 2005.

[144] Y. Zhang, Y. Sun, P. Phillips, G. Liu, X. Zhou, and S. Wang. A multilayer perceptron based smart pathological brain detection system by fractional fourier entropy. *Journal of medical systems*, 40(7):173, 2016.

[145] Y. Zhou, W. Su, L. Ding, H. Luo, and P. E. Love. Predicting safety risks in deep foundation pits in subway infrastructure projects: Support vector machine approach. *Journal of Computing in Civil Engineering*, 31(5):04017052, 2017.

[146] X. Zhu, X. Wu, and Y. Yang. Error detection and impact-sensitive instance ranking in noisy datasets. In *AAAI*, pages 378–384, 2004.